Lecture Notes in Computer Science Edited by G. Goos, J. Hartmanis and J. van Leeuwen

1666

Springer Berlin

Berlin Heidelberg New York Barcelona Hong Kong London Milan Paris Singapore Tokyo

Michael Wiener (Ed.)

Advances in Cryptology – CRYPTO '99

19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15-19, 1999 Proceedings



Series Editors

Gerhard Goos, Karlsruhe University, Germany Juris Hartmanis, Cornell University, NY, USA Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editor

Michael Wiener Entrust Technologies 750 Heron Road, Ottawa, Ontario, Canada KIV IA7 E-mail: michael.wiener@entrust.com

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Advances in cryptology: proceedings / CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15 - 19, 1999. Michael Wiener (ed.). [IACR]. - Berlin; Heidelberg; New York; Barcelona; Hong Kong; London; Milan; Paris; Singapore; Tokyo: Springer, 1999 (Lecture notes in computer science; Vol. 1666)
ISBN 3-540-66347-9

CR Subject Classification (1998): E.3, G.2.1, D.4.6, K.6.5, F.2.1-2, C.2, J.1

ISSN 0302-9743 ISBN 3-540-66347-9 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1999 Printed in Germany

Typesetting: Camera-ready by author

SPIN: 10704321 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

Crypto '99, the Nineteenth Annual Crypto Conference, was sponsored by the International Association for Cryptologic Research (IACR), in cooperation with the IEEE Computer Society Technical Committee on Security and Privacy and the Computer Science Department, University of California, Santa Barbara (UCSB). The General Chair, Donald Beaver, was responsible for local organization and registration.

The Program Committee considered 167 papers and selected 38 for presentation. This year's conference program also included two invited lectures. I was pleased to include in the program Ueli Maurer's presentation "Information-Theoretic Cryptography" and Martin Hellman's presentation "The Evolution of Public-Key Cryptography." The program also incorporated the traditional Rump Session for informal short presentations of new results, run by Stuart Haber.

These proceedings include the revised versions of the 38 papers accepted by the Program Committee. These papers were selected from all the submissions to the conference based on originality, quality, and relevance to the field of cryptology. Revisions were not checked, and the authors bear full responsibility for the contents of their papers.

The selection of papers was a difficult and challenging task. I wish to thank the Program Committee members who did an excellent job in reviewing papers and providing useful feedback to authors. Each submission was refereed by at least three reviewers. The Program Committee was assisted by many colleagues who reviewed submissions in their areas of expertise. My thanks go to all of them. External Reviewers included Michel Abdalla, Masayuki Abe, Bill Aiello, Kazumaro Aoki, Olivier Baudron, Don Beaver, Josh Benaloh, John Black, Simon Blackburn, Carlo Dan Boneh, Johan Borst, Antoon Bosselaers, Christian Cachin, Camenisch, Ran Canetti, Suresh Chari, Jean-Sébastien Coron, Janos Csirik, Erik De Win, Giovanni Di Crescenzo, Serge Fehr, Matthias Fitzi, Matt Franklin, Atsushi Fujioka, Juan Garay, Louis Granboulan, Shai Halevi, Héléna Handschuh, Kim Harrison, Martin Hirt, Russell Impagliazzo, Markus Jakobsson, Mariusz Jakubowski, Thomas Johansson, Marc Joye, Ari Juels, Charanjit Jutla, Burt Kaliski, Masayuki Kanda, Olaf Keller, Kunio Kobayashi, Tetsutaro Kobayashi, Ted Krovetz, Eyal Kushilevitz, Yue Lai, Susan Langford, Yishay Mansour, Keith Martin, Jim Massey, Phil MacKenzie, Andrew Mertz, Markus Michels, Victor Miller, Shiho Moriai, David Naccache, Moni Naor, Phong Nguyen, Tatsuaki Okamoto, Carles Padró, Pascal Paillier, Benny Pinkas, David Pointcheval, Guillaume Poupard, Vincent Rijmen, Kazue Sako, Kouichi Sakurai, Louis Salvail, Berry Schoenmakers, Nigel Smart, Jessica Staddon, Jacques Stern, Julien P. Stern, Douglas Stinson, Stuart Stubblebine, Youici Takasima, Keisuke Tanaka, Shigenori Uchiyama, Salil Vadhan, Ramarathnam Venkatesan, Ruizhong Wei, Avishai Wool, Yacov Yacobi, Lisa Yin, and Adam Young. I apologize for any inadvertent omissions.

The practice of accepting submissions electronically was continued for Crypto '99. Authors chose the electronic submission option for all but four papers. All credit for

VI Preface

the smooth electronic submission process goes to Joe Kilian, who handled all aspects and delivered a convenient directory full of submissions.

In organizing the scientific program and putting together these proceedings, I have been assisted by many people in addition to those mentioned above. In particular, I'd like to thank: Hugo Krawczyk, the Program Chair for Crypto '98, for his good advice and patience in answering my many questions; Don Coppersmith for his help throughout the review process; Donald Beaver, the General Chair of the conference, for freeing me from all issues not directly related to the scientific program and proceedings; Serge Mister for editing postscript submissions so that they would view and print acceptably; and Debbie Morton for secretarial help, particularly in helping to organize the Program Committee meeting.

Finally, I wish to thank all the authors who submitted papers, making this conference possible, and the authors of accepted papers for updating their papers in a timely fashion, allowing the production of these proceedings.

June 1999 Michael J. Wiener
Program Chair

CRYPTO '99

August 15-19, 1999, Santa Barbara, California, USA

Sponsored by the

International Association for Cryptologic Research (IACR)

in cooperation with

IEEE Computer Society Technical Committee on Security and Privacy, Computer Science Department, University of California, Santa Barbara

General Chair

Donald Beaver, CertCo, USA

Program Chair

Michael J. Wiener, Entrust Technologies, Canada

Program Committee

	Bell Laboratories, USA
Don Coppersmith	IBM Research, USA
Ivan Damgård	
Ronald Cramer	ETH Zurich, Switzerland
	IBM Research, USA
Andrew Klapper	University of Kentucky, USA
Lars Knudsen	University of Bergen, Norway
Xuejia Lai	r ³ security engineering, Switzerland
Andrew Odlyzko	AT&T Labs - Research, USA
	NTT Lab., Japan
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Jean-Jacques Quisquater	Université Catholique de Louvain, Belgium
Matt Robshaw	RSA Laboratories, USA
Phillip Rogaway	University of California at Davis, USA
Daniel Simon	Microsoft Research, USA
Serge Vaudenay	Ecole Normale Supérieure, France
Moti Yung	
	16. 1

Advisory Members

Mihir Bellare (Crypto 2000 Program Chair). Universit	ty of California at San Diego, USA
Joe Kilian (Electronic submissions)	NEC Research Institute, USA
Hugo Krawczyk (Crypto '98 Program Chair)	Technion, Israel and IBM, USA

Table of Contents

Public-Key Cryptanalysis I

On the Security of RSA Padding
Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization
The Hardness of the Hidden Subset Sum Problem and Its Cryptographic Implications
Invited Lecture
Information-Theoretic Cryptography
Secure Communication and Computation
Information Theoretically Secure Communication in the Limited Storage Space Model
The All-or-Nothing Nature of Two-Party Secure Computation
Distributed Cryptography
Adaptive Security for Threshold Cryptosystems
Two Party RSA Key Generation
Robust Distributed Multiplication without Interaction

A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting	. 148
Berry Schoenmakers	
Secret-Key Cryptography	
Truncated Differentials and Skipjack	. 165
Fast Correlation Attacks Based on Turbo Code Techniques	. 181
Highly Nonlinear Resilient Functions Optimizing Siegenthaler's Inequality	. 198
Message Authentication Codes	
UMAC: Fast and Secure Message Authentication	. 216
Square Hash: Fast Message Authentication via Optimized Universal Hash Functions	. 234
Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions	. 252
Stateless Evaluation of Pseudorandom Functions: Security Beyond the Birthday Barrier	. 270
Public-Key Cryptanalysis II	
Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97	. 288
Weakness in Quaternion Signatures	. 305
Cryptanalysis of "2R" Schemes	. 315

Factoring $N = p'q$ for Large r	326
Traitor Tracing	
An Efficient Public Key Traitor Tracing Scheme	38
Dynamic Traitor Tracing	354
Efficient Methods for Integrating Traceability and Broadcast Encryption	372
Differential Power Analysis	
Differential Power Analysis	388
Towards Sound Approaches to Counteract Power-Analysis Attacks	198
Signature Schemes	
Separability and Efficiency for Generic Group Signature Schemes	13
A Forward-Secure Digital Signature Scheme	131
Abuse-Free Optimistic Contract Signing	49
Zero Knowledge	
Can Statistical Zero Knowledge Be Made Non-interactive? or On the Relationship of <i>SZK</i> and <i>NISZK</i>	l67
On Concurrent Zero-Knowledge with Pre-processing	185

Asymmetric Encryption

On the Security Properties of OAEP as an All-or-Nothing Transform
Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization
Secure Integration of Asymmetric and Symmetric Encryption Schemes
Electronic Cash
Auditable, Anonymous Electronic Cash
Protocols and Broadcasting
Oblivious Transfer with Adaptive Queries
Compressing Cryptographic Resources
Coding Constructions for Blacklisting Problems without Computational Assumptions
Ravi Kumar, Sridhar Rajagopalan, Amit Sahai
An Information Theoretic Analysis of Rooted-Tree Based Secure Multicast Key Distribution Schemes
Author Index 639

On the Security of RSA Padding

Jean-Sébastien Coron¹⁺² David Naccache² Julien P. Stern³⁺⁴

- École Normale Supérieure 45 rue d'Ulm F-75005, Paris, France coron@clipper.ens.fr
- 2. Gemplus Card International 34 rue Guynemer Issy-les-Moulineaux, F-92447, France {coron,naccache}@gemplus.com
- 3. UCL Cryptography Group Bâtiment Maxwell, place du Levant 3 Louvain-la-Neuve, B-1348, Belgium stern@dice.ucl.ac.be
- 4. Université de Paris-Sud Laboratoire de Recherche en Informatique Bâtiment 490, F-91405, Orsay, France stern@lri.fr

Abstract. This paper presents a new signature forgery strategy.

The attack is a sophisticated variant of Desmedt-Odlyzko's method [11] where the attacker obtains the signatures of $m_1, \ldots, m_{\tau-1}$ and exhibits the signature of an m_{τ} which was never submitted to the signer; we assume that all messages are padded by a redundancy function μ before being signed.

Before interacting with the signer, the attacker selects τ smooth¹ $\mu(m_i)$ -values and expresses $\mu(m_{\tau})$ as a multiplicative combination of the padded strings $\mu(m_1), \ldots, \mu(m_{\tau-1})$. The signature of m_{τ} is then forged using the homomorphic property of RSA.

A padding format that differs from ISO 9796-1 by one single bit was broken experimentally (we emphasize that we could not extend our attack to ISO 9796-1); for ISO 9796-2 the attack is more demanding but still much more efficient than collision-search or factoring.

For DIN NI-17.4, PKCS #1 v2.0 and SSL-3.02, the attack is only theoretical since it only applies to specific moduli and happens to be less efficient than factoring; therefore, the attack does not endanger any of these standards.

1 Introduction

At a recent count (http://www.rsa.com), over 300 million RSA-enabled products had been shipped worldwide. This popularity, and the ongoing standardizations of signature and encryption formats [2,13,20,21,22,36] highlight the need to challenge claims that such standards eradicate RSA's multiplicative properties.

¹ an integer is ℓ -smooth if it has no bigger factors than ℓ .

Exponentiation is homomorphic and RSA-based protocols are traditionally protected against chosen-plaintext forgeries [9,11,35] by using a padding (or redundancy) function μ to make sure that :

$$RSA(\mu(x)) \times RSA(\mu(y)) \neq RSA(\mu(x \times y)) \mod n$$

In general, $\mu(x)$ hashes x and concatenates its digest to pre-defined strings; in some cases, substitution and permutation are used as well.

While most padding schemes gain progressive recognition as time goes by, several specific results exist: a few functions were broken by ad-hoc analysis ([16,24] showed, for instance, that homomorphic dependencies can still appear in $\mu(m) = a \times m + b$) while at the other extreme, assuming that the underlying building-blocks are ideal, some functions [5,6] are provably secure in the random oracle model.

The contribution of this paper is that the complexity of forging chosen message-signature pairs is sometimes much lower than that of breaking RSA $\circ \mu$ by frontal attacks (factoring and collision-search). The strategy introduced in this article does not challenge RSA's traditional security assumptions; instead, it seeks for multiplicative relations using the expected smoothness of moderate-size integers (the technique is similar in this respect to the quadratic sieve [33], the number field sieve [32] and the index-calculus method for computing discrete logarithm [1]).

As usual, our playground will be a setting in which the attacker $\mathcal A$ and the signer $\mathcal S$ interact as follows :

- \mathcal{A} asks \mathcal{S} to provide the signatures of $\tau 1$ chosen messages (τ being polylogarithmically-bounded in n). \mathcal{S} will, of course, correctly pad all the plaintexts before raising them to his secret power d.
- After the query phase and some post-processing, \mathcal{A} must exhibit the signature of at least one message (m_{τ}) which has never been submitted to \mathcal{S} .

Previous work : Misarsky's PKC'98 invited survey [30] is probably the best documented reference on multiplicative RSA forgeries. Davida's observation [9] is the basis of most RSA forgery techniques. [16,24] forge signatures that are similar to PKCs #1 v2.0 but do not produce their necessary SHA/MD5 digests [31,34]. [15] analyzes the security of RSA signatures in an interactive context. Michels et al. [28] create relations between the exponents of de Jonge-Chaum and Boyd's schemes; their technique extends to blind-RSA but does not apply to any of the padding schemes attacked in this paper. Baudron and Stern [4] apply lattice reduction to analyze the security of RSA $\circ \mu$ in a security-proof perspective.

A Desmedt-Odlyzko variant [11] applicable to padded RSA signatures is sketched in section 3.5 of [30]. It consists in factoring $\mu(m_{\tau})$ into small primes and obtaining the e-th roots of these primes from multiplicative combinations of signatures of messages which $\mu(m_i)$ -values are smooth. The signature of m_{τ} is forged by multiplying the e-th roots of the factors of $\mu(m_{\tau})$. The complexity of

this attack depends on the size of μ and not on the size of n; the approach is thus inapplicable to padding formats having the modulus' size (e.g. ISO 9796-2). In this paper we extend this strategy to padding schemes for which a linear combination of n and the padded value is small; when applied to William's scheme our attack allows to factor n.

2 A General Outline

Let $\{n,e\}$ be an RSA public key and d be the corresponding secret key. Although in this paper μ will alternatively denote ISO 9796-2, PKCS #1 V2.0, ANSI X9.31, SSL-3.02 or an ISO 9796-1 variant denoted \mathcal{F} , we will start by describing our attack in a simpler scenario where μ is SHA-1 or MD5 (in other words, messages will only be hashed before being exponentiated); the attack will be later adapted to the different padding standards mentioned above.

The outline of our idea is the following: since $\mu(m)$ is rather short (128 or 160 bits), the probability that $\mu(m)$ is ℓ -smooth (for a reasonably small ℓ) is small but non-negligible; consequently, if \mathcal{A} can obtain the signatures of chosen smooth $\mu(m_i)$ -values, then he could look for a message m_{τ} such that $\mu(m_{\tau})$ has no bigger factors than p_k (the k-th prime) and construct $\mu(m_{\tau})^d \mod n$ as a multiplicative combination of the signatures of the chosen plaintexts $m_1, \ldots, m_{\tau-1}$.

The difficulty of finding ℓ -smooth digests is a function of ℓ and the size of $\mu(m)$. Defining $\psi(x,y)=\#\{v< x, \text{ such that } v \text{ is } y\text{-smooth}\}$, it is known [12,14,19] that, for large x, the ratio $\psi(x,\sqrt[t]{x})/x$ is equivalent to Dickman's function defined by :

$$\rho(t) = \begin{cases} 1 & \text{if } 0 \le t \le 1\\ \rho(n) - \int_n^t \frac{\rho(v-1)}{v} dv & \text{if } n \le t \le n+1 \end{cases}$$

 $\rho(t)$ is thus an approximation of the probability that a *u*-bit number is $2^{u/t}$ -smooth; since $\rho(t)$ is somewhat cumbersome to compute, we refer the reader to appendix A for a lookup table.

Before we proceed, let us illustrate the concerned orders of magnitude. Referring to appendix A, we see that the probability that SHA/MD5 digests are 2^{24} -smooth is rather high ($\cong 2^{-19}, 2^{-13}$); this means that finding smooth digests would be practically feasible. This was confirmed by extensive simulations as illustrated by:

$${
m MD5(message~30854339~successfully~forged)} = 955dd317dd4715d26465081e4bfac000_{16} = 955dd317dd4715d26465081e4bfac000_{16}$$

 $2^{14} \times 3 \times 5^3 \times 13 \times 227 \times 1499 \times 1789 \times 2441 \times 4673 \times 4691 \times 9109 \times 8377619$

Several heuristics can, of course, accelerate the search: in our experiments, we factored only digests beginning or ending by a few zeroes; the optimal number of zeroes being a function of the running times of the attacker's hashing and factorization algorithms (parallelization is also possible).

In any case, denoting by L the size of the digest and by F(L) the factoring cost, the complexity of finding p_k -smooth digests is:

$$C_{L,k} = \mathcal{O}(\frac{F(L)}{\rho(L/\log_2(p_k))}) = \mathcal{O}(\frac{kL\log_2(p_k)}{\rho(L/\log_2(p_k))}) = \mathcal{O}(\frac{kL\log_2(k\ln k)}{\rho(L/\log_2(k\ln k))})$$

this is justified by the fact that p_k -smooth L-bit digests are expected only once per $1/\rho(L/\log_2(p_k))$ and that the most straightforward way to factor L is k trial divisions by the first primes (where each division costs $L\log_2(p_i)$ bit-operations).

These formulae should, however, be handled with extreme caution for the following reasons :

- \bullet Although in complexity terms L can be analyzed as a variable, one should constantly keep in mind that L is a fixed value because the output size of *specific* hash functions is not extensible.
- Trial division is definitely not the best candidate for F(L). In practice, our program used the following strategy to detect the small factors of $\mu(m)$: since very small divisors are very common, it is worthwhile attempting trial and error division up to $p_i \cong 2048$ before applying a primality test to $\mu(m)$ (the candidate is of course rejected if the test fails). As a next step, trial and error division by primes smaller than 15,000 is performed and the resulting number is handed-over to Pollard-Brent's algorithm [7] which is very good at finding small factors. Since it costs $\mathcal{O}(\sqrt{p_i})$ to pull-out p_i using Pollard-Brent's method we can further bound F(L) by $L\sqrt{p_k}$ to obtain:

$$C_{L,k} = \mathcal{O}(\frac{L\sqrt{k \ln k}}{\rho(L/\log_2(k \ln k))})$$

3 The Attack

The attack applies to RSA and Williams' scheme [37]; we assume that the reader is familiar with RSA but briefly recall Williams' scheme, denoting by J(x), the Jacobi symbol of x with respect to n.

In Williams' scheme $\mu(m) = 6 \mod 16$ and :

$$p = 3 \mod 8$$
 $e = 2$
 $q = 7 \mod 8$ $d = (n - p - q + 5)/8$

Before signing, S must check that $J(\mu(m)) = 1$. If $J(\mu(m)) = -1$, $\mu(m)$ is replaced by $\mu(m)/2$ to guarantee that $J(\mu(m)) = 1$ since J(2) = -1.

A signature s is valid if $w = s^2 \mod n$ is such that :

$$\mu(m) \stackrel{?}{=} \begin{cases} w & \text{if} \quad w = 6 \mod 8 \\ 2w & \text{if} \quad w = 3 \mod 8 \\ n - w & \text{if} \quad w = 7 \mod 8 \\ 2(n - w) & \text{if} \quad w = 2 \mod 8 \end{cases}$$

3.1 Finding Homomorphic Dependencies

The attack's details slightly differ between the RSA and Williams' scheme. For RSA, $\tau - 1$ chosen signatures will yield an additional $\mu(m_{\tau})^d \mod n$ while in Williams' case, τ chosen signatures will factor n. All chosen messages have the property that there exists a linear combination of $\mu(m_i)$ and n such that:

$$a_i \times n - b_i \times \mu(m_i)$$
 is p_k -smooth

where b_i is p_k -smooth as well.

It follows that $\mu(m_i)$ is the modular product of small primes:

$$\mu(m_i) = \prod_{j=1}^k p_j^{v_{i,j}} \bmod n \text{ for } 1 \le i \le \tau$$

Let us associate to each $\mu(m_i)$ a k-dimensional vector \mathbf{V}_i with coordinates $v_{i,j}$ taken modulo the public exponent e:

$$\mu(m_i) \longmapsto \mathbf{V}_i = \{v_{i,1} \bmod e, \dots, v_{i,k} \bmod e\}$$

We can now express, by Gaussian elimination, one of these vectors (reindexed as V_{τ}) as a linear combination of the others :

$$V_{\tau} = \sum_{i=1}^{\tau-1} \beta_i V_i \mod e, \quad \text{with} \quad \beta_i \in \mathbb{Z}_e$$
 (1)

From equation (1) we get:

$$v_{\tau,j} = \sum_{i=1}^{\tau-1} \beta_i v_{i,j} - \gamma_j \times e \text{ for all } 1 \le j \le k$$

and denoting $x = \prod_{j=1}^{k} p_j^{-\gamma_j}$:

$$\mu(m_{\tau}) = x^e \times \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i} \bmod n$$

For RSA, the forger will submit the $\tau-1$ first messages to $\mathcal S$ and forge the signature of m_τ by :

.

$$\mu(m_{\tau})^{d} = x \times \prod_{i=1}^{\tau-1} \left(\mu(m_{i})^{d} \right)^{\beta_{i}} \bmod n$$

In Williams' case, the signature of m_{τ} will be computed from the other signatures using equation (2) if J(x) = 1, using the fact that:

$$u = x^{2d} \bmod n = \begin{cases} x & \text{if } x \text{ is a square modulo } n \\ -x & \text{if not.} \end{cases}$$
$$\mu(m_{\tau})^{d} = \pm x \times \prod_{i=1}^{\tau-1} \left(\mu(m_{i})^{d}\right)^{\beta_{i}} \bmod n \tag{2}$$

If J(x) = -1, then $u^2 = x^2 \mod n$ and $(u - x)(u + x) = 0 \mod n$. Since J(x) = -J(u) we have $x \neq \pm u \mod n$ and GCD(u - x, n) will factor n. A can thus submit the τ messages to S, recover u, factor n and sign any message.

3.2 Expected Complexity

It remains, however, to estimate τ as a function of k:

- In the most simple setting e is prime and the set of vectors with k coordinates over \mathbb{Z}_e is a k-dimensional linear space; $\tau = k+1$ vectors are consequently sufficient to guarantee that (at least) one of the vectors can be expressed as a linear combination (easily found by Gaussian elimination) of the other vectors.
- When e is the r-th power of a prime p, $\tau = k+1$ vectors are again sufficient to ensure that (at least) one vector can be expressed as a linear combination of the others. Using the p-adic expansion of the vectors' coefficients and Gaussian elimination on k+1 vectors, we can write one of the vectors as a linear combination of the others.
 - Finally, the previous argument can be extended to the most general case:

$$e = \prod_{i=1}^{\omega} p_i^{r_i}$$

where it appears that $\tau = 1 + \omega k = \mathcal{O}(k \log e)$ vectors are sufficient to guarantee that (at least) one vector is a linear combination of the others; modulo each of the $p_i^{r_i}$, the attacker can find a set T_i of $(\omega - 1)k + 1$ vectors, each of which can be expressed by Gaussian elimination as a linear combination of k other vectors. Intersecting the T_i and using Chinese remaindering, one gets that (at least) one vector must be a linear combination of the others modulo e.

The overall complexity of our attack can therefore be bounded by:

$$C'_{L,k} = \mathcal{O}(\tau C_{L,k}) = \mathcal{O}(\frac{Lk \log e\sqrt{k \ln k}}{\rho(L/\log_2(k \ln k))})$$

and the attacker can optimize his resources by operating at a k where $C'_{L,k}$ is minimal.

Space complexity (dominated by the Gaussian elimination) is $\mathcal{O}(k^2 \log^3 e)$.

4 Analyzing Different Signature Formats

4.1 The Security of ISO/IEC-9796-1-like Signatures

ISO/IEC-9796-1 [21] was published in 1991 by ISO as the first international standard for digital signatures. It specifies padding formats applicable to algorithms providing message recovery (algorithms are not explicit but map r bits to r bits). ISO 9796-1 is not hashing-based and there are apparently no attacks [16,18] other than factoring on this scheme ([30]: "...ISO 9796-1 remains beyond the reach of all multiplicative attacks known today..."). The scheme is used to sign messages of limited length and works as follows when n and m are respectively $N=2\gamma+1$ and γ -bit numbers and $\gamma=4\ell$ is a multiple of eight.

Define by $a \cdot b$ the concatenation of a and b, let ω_i be the i-th nibble of m and denote by s(x) the hexadecimal substitution table²:

x =	0	1	2	3	4	5	6	7	8	9	A	В	C	D	Ε	F
s(x) =	Ε	3	5	8	9	4	2	F	0	D	В	6	7	A	С	1

Letting $\bar{s}(x)$ force the most significant bit in s(x) to 1 and $\tilde{s}(x)$ complement the least significant bit of s(x), ISO 9796-1 specifies:

$$\mu(m) = \bar{s}(\omega_{\ell-1}) \cdot \tilde{s}(\omega_{\ell-2}) \cdot \omega_{\ell-1} \cdot \omega_{\ell-2} \cdot s(\omega_{\ell-3}) \cdot s(\omega_{\ell-4}) \cdot \omega_{\ell-3} \cdot \omega_{\ell-4} \cdot \cdots s(\omega_3) \cdot s(\omega_2) \cdot \omega_3 \cdot \omega_2 \cdot s(\omega_1) \cdot s(\omega_0) \cdot \omega_0 \cdot 6_{16}$$

The attack that we are about to describe applies to a slight variant of ISO 9796-1 where $\tilde{s}(x)$ is replaced by s(x); this variant (denoted \mathcal{F}) differs from ISO 9796-1 by one single bit.

Let a_i denote nibbles and consider messages of the form :

$$m_i = a_6 \cdot a_5 \cdot a_4 \cdot a_3 \cdot a_2 \cdot a_1 \cdot 66_{16} \cdot a_6 \cdot a_5 \cdot a_4 \cdot a_3 \cdot a_2 \cdot a_1 \cdot 66_{16} \cdot \dots \cdot a_6 \cdot a_5 \cdot a_4 \cdot a_3 \cdot a_2 \cdot a_1 \cdot 66_{16}$$

which \mathcal{F} -padding is:

$$\mu(m_i) = \bar{s}(a_6) \cdot s(a_5) \cdot a_6 \cdot a_5 \cdot s(a_4) \cdot s(a_3) \cdot a_4 \cdot a_3 \cdot s(a_2) \cdot s(a_1) \cdot a_2 \cdot a_1 \cdot 2_{16} \cdot 2_{16} \cdot 6_{16} \cdot 6_{16} \cdot \dots \cdot s(a_6) \cdot s(a_5) \cdot a_6 \cdot a_5 \cdot s(a_4) \cdot s(a_3) \cdot a_4 \cdot a_3 \cdot s(a_2) \cdot s(a_1) \cdot a_2 \cdot a_1 \cdot 2_{16} \cdot 2_{16} \cdot 6_{16} \cdot 6_{16}$$

² actually, the bits of s(x) are respectively $\overline{x_3} \oplus x_1 \oplus x_0$, $\overline{x_3} \oplus x_2 \oplus x_0$, $\overline{x_3} \oplus x_2 \oplus x_1$ and $x_2 \oplus x_1 \oplus x_0$ but this has no importance in our analysis.

Restricting the choice of a_6 to the (eight) nibbles for which $s=\bar{s}$, we can generate 2^{23} numbers of the form $\mu(m_i)=x\times \Gamma_{23}$ where x is the 8-byte number $s(a_6)\cdot s(a_5)\cdot a_6\cdot a_5\cdot s(a_4)\cdot s(a_3)\cdot a_4\cdot a_3\cdot s(a_2)\cdot s(a_1)\cdot a_2\cdot a_1\cdot 2266_{16}$ and :

$$\Gamma_{23} = \sum_{i=0}^{\gamma/32-1} 2^{64i}$$

Section 3 could thus apply (treat Γ_{23} as an extra p_i) as soon as the expectation of p_k -smooth x-values reaches k+1:

$$k + 1 \sim 2^{23} \times \rho \left(\frac{64}{\log_2(k \ln k)} \right) \tag{3}$$

Using k = 3000 we forged thousands of 1024-bit \mathcal{F} -signatures in less than a day on a Pentium-PC (an example is given in appendix C). The attack is applicable to any $(64 \times c + 1)$ -bit modulus and its complexity is independent of $c \in \mathbb{N}$ (once computed, the same x-strings work with any such n).

k	# of p_k -smooth x -values (amongst 2^{23})	forgeries
345	346	1
500	799	298
1000	3203	2202
1500	6198	4697
2000	9344	7343
2500	12555	10054
3000	15830	12829

Table 1. Experimental \mathcal{F} -forgeries for 64-bit x-values, prime e.

The attack is equally applicable to 32, 48, 80, 96 or 112-bit x-strings (which yield 7, 15, 31, 39 and 47-bit plaintext spaces); a combined attack, mixing x-strings of different types is also possible (this has the drawback of adding the unknowns $\Gamma_7, \Gamma_{15}, \ldots$ but improves the probability of finding p_k -smooth x-strings). Long plain-English messages ending by the letter f can be forged using a more technical approach sketched in appendix B (66₁₆ represents the ASCII character f). Note, as a mere curiosity, a slight ($\cong 11\%$) experimental deviation from formula (3) due to the non-uniform distribution of the x-strings (which most and least significant bits can never be long sequences of zeroes). Finally, since the powers of 2 and Γ_{23} are identical, one can use k chosen messages instead of k+1, packing $p_1=2$ and $p_{k+1}=\Gamma_{23}$ into the updated unknown $p_1=2\Gamma_{23}$.

Non-impact on ISO 9796-1: The authors could not extend the attack to ISO 9796-1 and it would be wrong to state that ISO 9796-1 is broken.

Note: When we first looked into the standard, we did not notice \tilde{s} and we are grateful to Peter Landrock and Jørgen Brandt for drawing our attention to that. It appears from our discussions with ISO/JTC1/SC27 that \tilde{s} (the alteration that codes the message-border) has also been introduced to prevent arithmetic operations on $\mu(m)$; further information on ISO 9796-1 and our attack on \mathcal{F} will be soon posted on http://www.iso.ch/jtc1/sc27.

4.2 The Security of ISO 9796-2 Signatures

ISO 9796-2 is a generic padding standard allowing total or partial message recovery. Hash-functions of different sizes are acceptable and parameter L (in the standard k_h) is consequently a variable. Section 5, note 4 of [22] recommends $64 \le L \le 80$ for total recovery (typically an ISO 10118-2 [23]) and $128 \le L \le 160$ for partial recovery.

Partial Message Recovery. For simplicity, assume that N, L and the size of m are all multiples of eight and that the hash function is known to both parties. The message $m = m[1] \cdot m[2]$ is separated into two parts where m[1] consists of the N-L-16 most significant bits of m and m[2] of all the remaining bits of m. The padding function is:

$$\mu(m) = 6A_{16} \cdot m[1] \cdot HASH(m) \cdot BC_{16}$$

and m[2] is transmitted in clear.

Dividing $(6A_{16} + 1) \times 2^N$ by n we obtain :

$$(6A_{16} + 1) \times 2^N = i \times n + r \text{ with } r < n < 2^N$$

$$n' = i \times n = 6A_{16} \times 2^{N} + (2^{N} - r) = 6A_{16} \cdot n'[1] \cdot n'[0]$$

where n' is N+7 bits long and n'[1] is N-L-16 bits long.

Setting m[1] = n'[1] we get :

$$t = i \times n - \mu(m) \times 2^8 = n'[0] - \mathrm{HASH}(m) \cdot \mathtt{BC00_{16}}$$

where the size of t is less than L + 16 bits.

The forger can thus modify m[2] (and therefore HASH(m)) until he gets a set of messages which t-values are p_k -smooth and express one such $\mu(m_{\tau})$ as a multiplicative combination of the others.

Note that the attack is again independent of the size of n (forging 1024-bit signatures is *not* harder than forging 512-bit ones) but, unlike our \mathcal{F} -attack, forged messages are specific to a given n and can not be recycled when attacking different moduli.

To optimize efforts, \mathcal{A} must use the k minimizing $C'_{L+16,k}$.

Although the optimal time complexities for L=160 and L=128 are lower than the birthday complexities of SHA and MD5 we consider that L=160 implementations are still reasonably secure.

$L = k_h$	optimal $\log_2 k$	$\log_2 ext{time}$	$\log_2 \mathbf{space}$
128	18	54	36
160	20	61	40

Table 2. Attacks on ISO 9796-2, small public exponent.

Total Message Recovery. Assuming again that the hash function is known to both parties, that N and L are multiples of eight and that the size of m is N-L-16, function μ is:

$$\mu(m) = 4A_{16} \cdot m \cdot HASH(m) \cdot BC_{16}$$

Let us separate $m=m[1]\cdot m[0]$ into two parts where m[0] consists of the ℓ least significant bits of m and m[1] of all the remaining bits of m and compute, as in the previous case, an i such that :

$$n' = i \times n = 4A_{16} \cdot n'[1] \cdot n'[0]$$

where n'[0] is $(L + \ell + 16)$ -bits long and $n'[1] \cdot n'[0]$ is N-bits long. Setting m[1] = n'[1] we get:

$$t = i \times n - \mu(m) \times 2^8 = n'[0] - m[0] \cdot \text{HASH}(m) \cdot \text{BCOO}_{16}$$

where the size of t is less than $L + \ell + 16$ bits.

 \mathcal{A} will thus modify m[0] (and therefore HASH(m)) as needed and conclude the attack as in the partial recovery case. ℓ must be tuned to expect just enough p_k -smooth t-values with a reasonably high probability i.e.:

$$k \sim 2^{\ell} \times \rho \left(\frac{L + \ell + 16}{\log_2(k \ln k)} \right)$$

The complexities summarized in the following table (a few PC-weeks for $k_h = 64$) seem to suggest a revision of this standard.

$L = k_h$	optimal $\log_2 k$	$\log_2 \mathbf{time}$	$\log_2 \mathbf{space}$	ℓ
64	15	47	30	32
80	17	51	34	34

Table 2 (continued). Attacks on ISO 9796-2, small public exponent.

Note that our attack would have applied as well to:

$$\mu(m) = \mathtt{4A_{16}} \cdot \mathrm{HASH}(m) \cdot m \cdot \mathtt{BC_{16}}$$

In which case take $n' = i \times n$ such that $n' \mod 256 = \mathtt{BC}_{16}$ and use m to replicate the least significant bits of n'; subtraction will then yield a moderate size integer times of a power of two.

An elegant protection against our attack is described in [13] (its security is basically comparable to that of PKCS #1 v2.0, discussed later on in this paper); a second efficient solution, suggested by Jean-Jacques Quisquater in the rump session of CRYPTO'97 is:

$$\mu(m) = \mathtt{4A_{16}} \cdot (m \oplus \mathrm{HASH}(m)) \cdot \mathrm{HASH}(m) \cdot \mathtt{BC_{16}}$$

4.3 Analyzing PKCS #1 V2.0, SSL-3.02 and ANSI X9.31

This section describes theoretical attacks on PKCS #1 V2.0, SSL-3.02 and ANSI x9.31 which are better than the birthday-paradox. Since our observations are not general (for they apply to moduli of the form $n = 2^k \pm c$) and more demanding than factorization, they do not endanger current implementations of these standards. It appears that $n = 2^k \pm c$ offers regular 1024-bit RSA security as far as c is not much smaller than 2^{500} , and square-free c-values as small as 400 bits may even be used [25]. In general $(n > 2^{512})$ such moduli appear to offer regular security as long as $\log_2(c) \cong \log_2(n)/2$ and c is square-free [26].

Although particular, $n=2^k\pm c$ has been advocated by a number of cryptographers for it allows trial and error divisions to be avoided. For instance, the informative annex of ISO 9796-1 recommends "...some forms of the modulus $(n=2^k\pm c)$ [that] simplify the modulo reduction and need less table storage.". Note however, that even in our worst scenario, ISO 9796-1's particular form is still secure: for 1024-bit moduli, ISO 9796-1 recommends a 767-bit c whereas our attack will require a 400-bit c. The reader is referred to section 14.3.4 of [27] for further references on $n=2^k\pm c$.

Assume that we are given a 1024-bit $n=2^k-c$, where $\ell=\log_2(c)\cong 400$ and c is square-free; we start by analyzing SSL-3.02 where :

$$\mu(m) = 0001_{16} \cdot \text{FFFF}_{16} \dots \text{FFFF}_{16} \cdot 00_{16} \cdot \text{SHA}(m) \cdot \text{MD5}(m)$$

 $n-2^{15} \times \mu(m)$ is an ℓ -bit number on which we conduct an ISO 9796-2-like attack which expected complexity is $C'_{\ell,k}$.

The characteristics of the attack are summarized in table 3 which should be compared to the birthday paradox (2^{144} time, negligible space) and the hardness of factorization ($\{\text{time, space}\}\$ denote the base-two logarithms of the time and space complexities of the attacks):

$\log_2 n$	ℓ	optimal	$\log_2 k$	our attack	factorization
606	303		28	$\{84, 56\}$	$\{68, 41\}$
640	320		29	$\{87, 58\}$	$\{70, 42\}$
768	384		33	$\{97, 66\}$	$\{75, 45\}$
1024	400		34	$\{99, 68\}$	$\{86, 50\}$
1024	512		39	{115, 78}	$\{86, 50\}$

Table 3. Estimates for SSL 3.02, small public exponent.

The phenomenon also scales-down to PKCS $\#1\ V2.0$ where:

$$\begin{split} \mu(m) &= 0001_{16} \cdot \text{FFFF}_{16} \dots \text{FFFF}_{16} \cdot 00_{16} \cdot c_{\text{SHA}} \cdot \text{SHA}(m) \\ \mu(m) &= 0001_{16} \cdot \text{FFFF}_{16} \dots \text{FFFF}_{16} \cdot 00_{16} \cdot c_{\text{MD5}} \cdot \text{MD5}(m) \\ c_{\text{SHA}} &= 302130090605280E03021A05000414_{16} \end{split}$$

 $c_{\mathrm{MD5}} = 3020300006082 \text{A}864886 \text{F}70 \text{D}020505000410}_{16}$

and:

$\log_2 n$	ℓ	optimal	$\log_2 k$	our attack	factorization
512	256		23	$\{77, 46\}$	$\{64, 39\}$
548	274		27	$\{80, 54\}$	$\{66, 40\}$

Table 4. Estimates for PKCS #1 V2.0 and ANSI X9.31, small public exponent.

These figures appear roughly equivalent to a birthday-attack on SHA, even for rather small (550-bit) moduli. Note that the attack applies to $n = 2^k + c$ by computing $n - 2^{14} \times \mu(m)$.

Note: In a recent correspondence, Burt Kaliski informed us that Ron Rivest developed in 1991 a forgery strategy which is a simple case of the one described in this paper; the design of PKCS #1 v1.5 took this into account, but Ron's observation was never published. Further information on our attack will appear soon in an RSA bulletin http://www.rsa.com/rsalabs/.

A similar analysis where the prescribed moduli begin by 6BBBBB... $_{16}$ is applicable to ANSI x9.31 (yielding exactly the same complexities as for PKCS #1 v2.0) where :

$$\mu(m) = 6 \mathtt{B}_{16} \cdot \mathtt{BBBB}_{16} \dots \mathtt{BBBB}_{16} \cdot \mathtt{BA}_{16} \cdot \mathtt{SHA}(m) \cdot \mathtt{33CC}_{16}$$

ANSI X9.31 recommends to avoid $n=2^k\pm c$. If one strictly follows the standard n=6BBBBB...₁₆ can not occur (the standard requires a bit length which is a multiple of eight) but one could in theory work with $2\mu(m)$ instead of $\mu(m)$.

Finally, we will consider a theoretical setting in which an authority certifies moduli generated by users who wish to join a network; naturally, users never reveal their secret keys but using storage optimizations as a pretext, the authority implements an ID-based scheme where different $random\ looking$ bits (registration ID, account numbers etc) are forced into the most significant bits of each n [26]. Users generate moduli having the prescribed patterns they receive.

If the authority can find two small constants $\{u, v\}$ such that :

$$\log_2(u \times n - v \times \mu(m)) \cong \eta$$
 for a moderate η (4)

then our attack would extend to moduli which are not necessarily of the form $2^k \pm c$. To do so, oversimplify the setting to $\mu(m) = (2^w - 1) \cdot f(m)$ and $n = n[1] \cdot n[0]$ where n[0] has the size of f(m) and substitute these definitions in equation (4):

$$\log_2(u\times (n[1]\cdot n[0]) - v\times ((2^w-1)\cdot f(m))) \cong \eta$$

since the authority has no control over f(m), the best thing to do would be to request that $u \times n[1] = v \times (2^w - 1)$ which results in an $\eta \cong \log_2(f(m)) + \log_2(\max\{u,v\})$.

The authority can thus prescribe moduli which most significant bits are $v_i \times (2^w - 1)/u_i$ where u_i are moderate-size factors of $2^w - 1$. Such factors look random and should not raise the user's suspicion.

We can therefore conclude that although practically safe, the use of authority-specified moduli in fixed-pattern padding contexts might be an interesting theoretical playground.

5 Conclusion and Further Research

Although the analysis presented in this paper indicates a weakness in ISO 9796-2 when $k_h \cong 64$, products using this standard should not be systematically withdrawn; a few product analyzes reveal that system-level specifications (message contents, insufficient access to \mathcal{S} etc.) frequently make real-life attacks harder than expected.

It seems reasonable (although we can not base our belief on formal grounds) that good message recovery padding schemes should be usable for encryption as well; we motivate this recommendation by the functional similarity between RSA encryption and message recovery.

Full-domain-hash offers the best possible protection against our attack and we advocate its systematic use whenever possible. If impossible, it seems appropriate to link L and N since for a fixed L there is necessarily a point (birthday) above which increasing N will slow-down the legitimate parties without improving security.

We also recommend four research directions:

- An integer is $\{a, p_k\}$ -semismooth [3] if each of its prime factors is smaller than a and all but one are smaller than p_k . A well known-strategy (called the large prime variant) consists of searching, using the birthday paradox, $\{a, p_k\}$ -semismooth $\{\mu(x), \mu(y)\}$ pairs having an identical large prime factor (e.g. 80-bits long); the ratio $\mu(x)/\mu(y)$ mod n can then be used as one p_k -smooth input in the Gaussian elimination.
- It might be interesting to find out if our \mathcal{F} -attack could handle \tilde{s} by using a different Γ :

$$\Gamma = \Delta \cdot 000000000001_{16} \cdot 00000000001_{16} \cdots 00000000001_{16}$$

In which case x-values should end by the pattern 2266₁₆, be p_k -smooth and such that $x' = x/\Delta$ is a valid message header. Note that different Δ -values might be mixed in the same attack, using a large prime variant where the different Γ -values are eliminated by modular division.

- Although we have no specific instances for the moment, one could also try to combine our technique with [4] to speed-up forgery in specific situations.
- Finally, it appears that incomplete ad-hoc analyzes of hash-functions (building digests with u prescribed bits in less than 2^u operations) could be the source of new problems in badly designed padding schemes.

Acknowledgments

We are grateful to Arjen Lenstra, Pascal Paillier and Michael Tunstall for their helpful comments, the authors also thank Pascal Autissier, Christophe Clavier, Renato Menicocci and Phong N'guyen for their insights into several mathematical details. Last but not least, we express our recognition to Burt Kaliski, Bart Preneel and Jean-Jacques Quisquater for their help.

References

- L. Adleman, A subexponential algorithm for the discrete logarithm problem with applications to cryptography, Proceedings of the IEEE 20-th Annual symposium on the foundations of computer science, pp. 55-60, 1979.
- ANSI X9.31, Digital signatures using reversible public-key cryptography for the financial services industry (rDSA), 1998.
- 3. E. Bach and R. Peralta, Asymptotic semismoothness probabilities, Mathematics of computation, vol. 65, no. 216, pp. 1701–1715, 1996.
- 4. O. Baudron and J. Stern, To pad or not to pad: does formatting degrade security?, 1999 RSA Data Security Conference proceeding book, 1999.
- M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, Proceedings of the first annual conference on computer and communication security, ACM, 1993.
- M. Bellare and P. Rogaway, The exact security of digital signatures: how to sign with RSA and Rabin, Advances in cryptology EUROCRYPT'96, Springer-Verlag, Lectures notes in computer science 1070, pp. 399–416, 1006
- R. Brent, An improved Monte Carlo factorization algorithm, Nordisk Tidskrift för Informationsbehandling (BIT) vol. 20, pp. 176–184, 1980.
- 8. N. de Bruijn, On the number of positive integers $\leq x$ and free of prime factors $\geq y$, Indagationes Mathematicae, vol. 13, pp. 50–60, 1951. (cf. as well to part II, vol. 28, pp. 236–247, 1966.).
- G. Davida, Chosen signature cryptanalysis of the RSA (MIT) public-key cryptosystem, TR-CS-82-2, Department of electrical engineering and computer science, University of Wisconsin, Milwaukee, 1982.
- D. Denning, Digital signatures with RSA and other public-key cryptosystems, Communications of the ACM, vol. 27-4, pp. 388–392, 1984.
- Y. Desmedt and A. Odlyzko. A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes, Advances in cryptology CRYPTO'85, Springer-Verlag, Lectures notes in computer science 218, pp. 516–522, 1986.
- K. Dickman, On the frequency of numbers containing prime factors of a certain relative magnitude, Arkiv för matematik, astronomi och fysik, vol. 22A, no. 10, pp. 1–14, 1930.

- DIN NI-17.4, Specification of chipcard interface with digital signature application/function according to SigG and SigV, version 1.0, 1998.
- J. Dixon, Asymptotically fast factorization of integers, Mathematics of computation, vol. 36, no. 153, pp. 255–260, 1981.
- J. Evertse and E. van Heyst, Which new RSA-signatures can be computed from certain given RSA signatures?, Journal of cryptology vol. 5, no. 1, 41–52, 1992.
- M. Girault, J.-F. Misarsky, Selective forgery of RSA signatures using redundancy, Advances in cryptology Eurocrypt'97, Springer-Verlag, Lectures notes in computer science 1233, pp. 495–507, 1997.
- J. Gordon, How to forge RSA key certificates, Electronic Letters, vol. 21, no. 9, April 25-th, 1985.
- L. Guillou, J.-J. Quisquater, M. Walker, P. Landrock and C. Shaer, Precautions taken against various attacks in ISO/IEC DIS 9796, Advances in cryptology Eurocrypt'90, Springer-Verlag, Lectures notes in computer science 473, pp. 465–473, 1991.
- H. Halberstam, On integers whose prime factors are small, Proceedings of the London mathematical society, vol. 3, no. 21, pp. 102–107, 1970.
- K. Hickman, The SSL Protocol, December 1995. Available electronically at: http://www.netscape.com/newsref/std/ssl.html
- ISO/IEC 9796, Information technology Security techniques Digital signature scheme giving message recovery, Part 1: Mechanisms using redundancy, 1999.
- ISO/IEC 9796-2, Information technology Security techniques Digital signature scheme giving message recovery, Part 2: Mechanisms using a hash-function, 1997.
- ISO/IEC 10118-2, Information technology Security techniques Hashfunctions; Part 2: Hash functions using an n-bit block-cipher algorithm, 1994.
- W. de Jonge and D. Chaum. Attacks on some RSA signatures, Advances in cryptology CRYPTO'85, Springer-Verlag, Lectures notes in computer science 218, pp. 18–27, 1986.
- A. Lenstra, Generating RSA moduli with a predetermined portion, Advances in cryptology ASIACRYPT'98, Springer-Verlag, Lectures notes in computer science 1514, pp. 1–10, 1998.
- A. Lenstra, de auditu, January 1999.
- 27. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of applied cryptography*, CRC Press.
- 28. M. Michels, M. Stadler and H.-M. Sun, On the security of some variants of the RSA signature scheme, Computer security-ESORICS'98, Springer-Verlag, Lectures notes in computer science 1485, pp. 85–96, 1998.
- J.-F. Misarsky, A multiplicative attack using LLL algorithm on RSA signatures with redundancy, Advances in cryptology CRYPTO'97, Springer-Verlag, Lectures notes in computer science 1294, pp. 221–234, 1997.

- J.-F. Misarsky, How (not) to design RSA signature schemes, Public-key cryptography, Springer-Verlag, Lectures notes in computer science 1431, pp. 14–28, 1998.
- National Institute of Standards and Technology, Secure hash standard, FIPS publication 180-1, April 1994.
- J. Pollard, Factoring with cubic integers, The development of the number field sieve, Springer-Verlag, Lectures notes in computer science 1554, pp. 4–10, 1993.
- C. Pomerance, The quadratic sieve factoring algorithm, Advances in cryptology Eurocrypt'84, Springer-Verlag, Lectures notes in computer science 209, pp. 169–182, 1985.
- R. Rivest, RFC 1321: The MD5 message-digest algorithm, Internet activities board, April 1992.
- R. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, vol. 21-2, pp. 120–126, 1978.
- RSA Laboratories, PKCS #1: RSA cryptography specifications, version 2.0, September 1998.
- H. Williams, A modification of the RSA public key encryption procedure, IEEE TIT, vol. 26, pp. 726–729, 1980.

APPENDIX A

The following (redundant) look-up table lists ρ for the various smoothness and digest-size values concerned by this paper; $\rho(136/24)$, the probability that a 136-bit number has no prime factors larger than 2^{24} is $2^{-14.2}$:

$-\log_2 \rho \searrow$	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72
32	1.7	0.9	0.5	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
48	4.4	2.7	1.7	1.1	0.8	0.5	0.3	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
64	7.7	5.0	3.4	2.4	1.7	1.2	0.9	0.7	0.5	0.3	0.2	0.0	0.0	0.0	0.0
80	11.5	7.7	5.4	3.9	2.9	2.2	1.7	1.3	1.0	0.8	0.6	0.5	0.4	0.3	0.2
96	15.6	10.7	7.7	5.7	4.4	3.4	2.7	2.1	1.7	1.4	1.1	0.9	0.8	0.6	0.5
112	20.1	13.9	10.2	7.7	5.9	4.7	3.8	3.1	2.5	2.1	1.7	1.4	1.2	1.0	0.8
128	24.9	17.4	12.8	9.8	7.7	6.1	5.0	4.1	3.4	2.8	2.4	2.0	1.7	1.4	1.2
136	27.4	19.2	14.2	10.9	8.6	6.9	5.6	4.6	3.9	3.2	2.8	2.3	2.0	1.7	1.5
144	29.9	21.1	15.6	12.0	9.5	7.7	6.3	5.2	4.4	3.7	3.1	2.7	2.3	2.0	1.7
152	32.4	22.9	17.1	13.2	10.5	8.5	7.0	5.8	4.9	4.1	3.5	3.0	2.6	2.3	2.0
160	35.1	24.9	18.6	14.4	11.5	9.3	7.7	6.4	5.4	4.6	3.9	3.4	2.9	2.6	2.2
168	37.9	26.9	20.1	15.6	12.5	10.2	8.4	7.0	5.9	5.1	4.4	3.8	3.3	2.9	2.5
176	40.6	28.9	21.7	16.9	13.5	11.0	9.1	7.7	6.5	5.6	4.8	4.2	3.6	3.2	2.8
400	129.	95.2	73.9	59.2	49.0	41.5	35.1	30.2	26.5	23.1	20.8	18.5	16.7	15.1	13.7
512	179.	133	104	84.0	69.8	59.0	50.8	44.0	38.8	34.1	30.6	27.2	24.9	22.5	20.6

The table uses the exact formula (section 2) for $t \leq 10$ and de Bruijn's approximation [8] for t > 10:

$$\rho(t) \cong (2\pi t)^{-1/2} \exp\left(\gamma - t\zeta + \int_0^{\zeta} \frac{e^s - 1}{s} ds\right)$$

where ζ is the positive solution of $e^{\zeta} - 1 = t\zeta$ and γ is Euler's constant.

APPENDIX B

The attack's time-consuming part is the exhaustive-search of k appropriate x-strings; therefore, when one wants the x-strings to be 256-bits long, the increase in k makes the attack impractical.

To overcome this problem, we suggest the following: as a first step, collect the signatures corresponding to moderate-size p_k -smooth x-strings (which are relatively easy to find) and extract from their appropriate multiplicative combinations the e-th roots of the k first primes. Then, exhaustive-search two plain-English 128-bit messages $\{m, m'\}$ ending by the letter f such that $\mu(m)/\Gamma$ and $\mu(m')/\Gamma$ are both p_k -smooth, with:

$$\Gamma = 2^{256(c-1)} + \ldots + 2^{256} + 1$$

for a $(256 \times c + 1)$ -bit modulus. Since we only need two such numbers, the overall workload is very tolerable. Next, submit m to S and divide its signature by the e-th roots of its small prime factors to recover $\Gamma^d \mod n$. Using $\Gamma^d \mod n$ and the e-th roots of the k first primes we can now forge, by multiplication, the signature of m'.

APPENDIX C

This appendix contains an \mathcal{F} forgery that works with any 1025-bit modulus; to fit into the appendix, the example was computed for e=3 but forgeries for other public exponents are as easy to obtain.

step 1 : Select any 1025-bit RSA modulus, generate $d=3^{-1} \mod \phi(n)$, let $\mu=\mathcal{F}$ and form the 180 messages :

$$m_i = (256 \times \mathtt{message}[i]_{16} + 102) \times \sum_{j=0}^{11} 2^{32j}$$

where message[i] denotes the elements of the following table:

00014E	008C87	00D1E8	01364B	0194D8	01C764	021864	03442F	0399FB	048D9E	073284	0863DE	09CCE8
0A132E	0A2143	0BD886	0C364A	0C368C	OC6BCF	OD3AC1	0D5C02	0EA131	0F3D68	0F9931	31826A	31BE81
31ED6B	31FCD0	320B25	32B659	332D04	3334D8	33EAFC	33EB1D	343B49	353D02	35454C	35A1A9	36189E
362C79	365174	3743AB	3765F6	37C1E2	3924AC	3998A8	3AF8A7	3B6900	ЗВ9ЕЕВ	3BC1FF	3DE2DE	3E51BE
3E8191	3F49F3	3F69AC	4099D9	40BF29	41C36C	41D8C0	424EE8	435DB7	446DC1	4499CC	44AA20	44EE53
4510E8	459041	45A464	45AA03	460B80	4771E7	486B6A	499D40	4A5CF8	4AC449	4ADAOA	4B87A8	4C06A1
4C5C17	4D4685	4E39EA	4EB6B6	4F8464	716729	71C7D3	71FA22	722209	72DBF1	7619AB	765082	767C39
76885C	78F5F3	79E412	79FAD6	7CD0ED	7DOABA	7DBA1D	7DE6A5	7E06A2	7EA5F2	7EC1ED	7EEC78	90BB4B
90DE38	9139D7	934C2C	9366C5	941809	941BFB	947EB4	94DB29	952D45	9745BD	978897	97A589	9827AF
984FAC	9A193D	9A83E2	9B74E3	9BEAE9	9C704F	9DBA98	9F9337	A00D15	A02E3D	A10370	A429A6	A4DADD
A4F689	A5485D	A6D728	A76B0F	A7B249	A87DF3	A95438	A96AA4	AB1A82	AD06A8	AEAODO	AEB113	D076C5
D13F0E	D18262	D1B0A7	D35504	D3D9D4	D3DEE4	D4F71B	D91C0B	D96865	DA3F44	DB76A8	DE2528	DE31DD
DE46B8	DE687D	DEB8C8	DF24C3	DFDFCF	DFF19A	E12FAA	E1DD15	E27EC1	E39C56	E40007	E58CC8	E63CE0
E6596C	E7831E	E796FB	E7E80C	E85927	E89243	E912B4	E9BFFF	EAODFC	EACF65	EB29FA		

step 2: construct the message $m' = \text{EE7E8E66}_{16} \times \sum_{j=0}^{11} 2^{32j}$ and obtain from the signer the 180 signatures $s_i = \mu(m_i)^d \mod n$.

step 3: the signature of m' is:

$$\mu(m')^d = \prod_{i=0}^{345} p_i^{-\texttt{gamma[}i\texttt{]}} \prod_{i=1}^{180} s_i^{\texttt{beta[}i\texttt{]}} \bmod n$$

where p_i denotes the *i*-th prime (with $p_0 = \Gamma_{23}$) and beta[*i*] denotes the elements of the following table :

1	2	1	2	2	2	2	1	2	2	2	1	1	2	2	2	1	1	2	1	2	2	2	2	2	1	1	2	1	1	2	1	1	2	1	1
1	1	1	1	1	1	2	1	1	1	1	2	1	1	1	1	2	2	1	1	2	1	2	1	1	2	2	1	1	1	1	2	1	1	2	1
1	1	1	1	2	2	1	2	1	2	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	2
1	1	1	2	2	2	2	1	2	2	1	1	2	2	2	2	1	1	2	1	2	2	2	2	1	1	1	2	1	1	2	1	1	1	1	2
2	1	1	1	1	2	2	1	2	2	1	1	2	2	2	1	2	1	2	2	2	1	2	1	2	2	1	2	1	2	2	1	1	2	1	1

gamma[i] represents the hexadecimal values:

57	57	68	33	27	18	16	13	10	0F	0E	OB	09	09	OD	05	ΟB	07	04	80	07	07	07	09	OA	03	07
04	05	05	03	04	03	01	02	03	04	03	01	03	03	03	02	06	03	03	04	06	02	04	04	02	02	03
02	04	04	03	04	01	04	03	02	03	02	01	02	02	01	03	01	01	01	01	03	03	01	03	02	02	01
04	02	04	02	02	01	02	01	01	01	03	03	01	02	01	01	00	03	02	03	01	01	02	01	02	02	03
03	04	03	03	02	03	01	02	03	02	01	03	02	02	01	01	00	02	01	01	03	01	01	01	01	01	02
00	02	00	00	01	02	01	01	01	00	01	01	00	01	01	02	02	01	01	01	00	01	00	01	01	04	02
02	02	01	02	02	01	02	01	02	00	01	00	02	01	02	02	00	01	02	01	01	01	02	01	01	01	02
01	00	01	01	00	00	01	02	00	01	00	01	01	00	01	00	01	02	02	01	01	02	00	00	02	01	02
02	01	00	00	01	00	01	00	01	00	02	00	00	00	01	01	00	00	01	01	00	00	00	01	00	00	00
00	00	00	01	01	00	00	01	02	01	01	01	00	01	02	01	01	01	02	00	00	00	01	01	00	01	00
00	00	02	02	01	00	01	02	00	01	00	01	02	00	01	00	00	01	00	01	01	01	00	01	01	00	01
01	01	01	00	00	01	01	00	00	01	01	00	01	01	00	00	01	00	00	00	01	01	02	02	01	01	00
00	01	02	01	02	00	01	01	00	01	00	00	00	00	00	00	01	00	00	01	02	01					

Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization

Aviad Kipnis¹ and Adi Shamir²

 $^{\rm 1}$ NDS Technologies, Israel $^{\rm 2}$ Computer Science Dept., The Weizmann Institute, Israel

Abstract. The RSA public key cryptosystem is based on a single modular equation in one variable. A natural generalization of this approach is to consider systems of several modular equations in several variables. In this paper we consider Patarin's Hidden Field Equations (HFE) scheme, which is believed to be one of the strongest schemes of this type. We represent the published system of multivariate polynomials by a single univariate polynomial of a special form over an extension field, and use it to reduce the cryptanalytic problem to a system of ϵm^2 quadratic equations in m variables over the extension field. Finally, we develop a new relinearization method for solving such systems for any constant $\epsilon > 0$ in expected polynomial time. The new type of attack is quite general, and in a companion paper we use it to attack other multivariate algebraic schemes, such as the Dragon encryption and signature schemes. However, we would like to emphasize that the polynomial time complexities may be infeasibly large for some choices of the parameters, and thus some variants of these schemes may remain practically unbroken in spite of the new attack.

1 Introduction

The problem of developing new public key encryption and signature schemes had occupied the cryptographic research community for the last 20 years. A particularly active line of research was based on the observation that solving systems of modular multivariate polynomial equations is NP-complete. Consider, for example, a public encryption key consisting of n random quadratic polynomials in n variables over the two element field $\mathbf{F_2}$. To encrypt an n bit cleartext, you assign each bit to a variable, and evaluate the n quadratic polynomials modulo 2. To decrypt this ciphertext, you use it as a right hand side and solve the resultant system of n quadratic equations in n unknowns. When n=100, the encryption process is extremely fast, while the decryption process (by an eavesdropper) seems to be completely infeasible.

The system of equations must contain a trapdoor, whose knowledge makes it possible to solve the system of equations efficiently for any right hand side. The main difference between the various multivariate schemes is the type of trapdoor structure they embed into the published polynomials.

An early example of a multivariate signature scheme was developed by Ong Schnorr and Shamir [OSS84], and was broken shortly afterwards by Pollard and Schnorr [PS87]. Fell and Diffie [FD85] published another multivariate scheme, but observed that it was insecure for any practical key size. A different type of trapdoor was developed by Matsumoto and Imai [MI88], but their scheme was shown to be insecure in Patarin [P95]. Shamir [S93] proposed two multivariate schemes modulo large n = pq, which were shown to be insecure by Coppersmith Stern and Vaudenay [CSV97]. In an attempt to revive the field, Patarin had developed several new types of trapdoors. The simplest of his new constructions was the Oil and Vinegar signature scheme [P97], which was broken by Kipnis and Shamir [KS98]. A more secure construction was the Dragon encryption and signature schemes, described in Patarin [P96b] and Koblitz [K98]. A simplified version of this scheme (called Little Dragon) was broken by Coppersmith and Patarin, but the original Dragon scheme remained unbroken. The Hidden Field Equations (HFE) was published in Patarin [P96a], and conjectured by its author to be the strongest among his various constructions. In spite of extensive cryptanalytic effort, no attacks on the HFE scheme had been published so far.

In this paper we develop a new cryptanalytic approach and use it to attack both the HFE scheme (as shown in this paper) and the Dragon scheme (as shown in a companion paper). The asymptotic complexity of the attack is polynomial (when some of the parameters grow to infinity while others are kept fixed), but the basic implementation described in this paper may be impractical for sufficiently large keys. Both the scheme and the attack can be enhanced in numerous ways, and thus it is too early to decide whether some variant of the HFE scheme can survive an optimized version of the attack.

The attack is based on the observation that any given system of n multivariate polynomials in n variables over a field \mathbf{F} can be represented by a single univariate polynomial of a special form over K which is an extension field of degree n over **F**. We analyze the effect of the trapdoor hiding operations on this representation, and use it in order to translate the original problem of solving nquadratic equations in n variables over the small field \mathbf{F} into a new problem of solving a system of ϵm^2 quadratic equations in m variables over the large field \mathbf{K} , where m is a small multiple of n. The standard linearization technique for solving such systems is to replace any product of variables $x_i x_j$ by a new variable y_{ij} , and to solve the resultant system of ϵm^2 linear equations in the $m^2/2$ new y_{ij} variables. However, in our attack $\epsilon < 0.5$, and thus the linearization technique creates exponentially many parasitic solutions which do not correspond to solutions of the original quadratic equations. We overcome this problem by developing a general new technique (called relinearization) which is expected to solve random systems of equations of this type in polynomial time for any fixed $\epsilon > 0$. Since no previously published technique could handle such systems, we expect the relinearization technique to have additional applications in cryptanalysis, algorithm design, and operations research.

2 The HFE Scheme

The HFE encryption algorithm was presented by Jacques Patarin at Eurocrypt '96. It uses a small field \mathbf{F} with q elements (the recommended choice is q=2), and a large extension field \mathbf{K} of degree n over \mathbf{F} (the recommended choice is n=128, yielding a field \mathbf{K} with 2^{128} elements). The field \mathbf{K} can be viewed as a vector space of dimension n over \mathbf{F} , and the mapping between the two representations is defined by a basis of n elements $\omega_0, \ldots, \omega_{n-1}$ in \mathbf{K} via $\sum_{i=0}^{n-1} x_i \omega_i \leftrightarrow (x_0, \ldots, x_{n-1})$.

To construct his public key, the user picks a random univariate polynomial P(x) over **K** of the form

$$P(x) = \sum_{i=0}^{r-1} \sum_{j=0}^{r-1} p_{ij} x^{q^i + q^j}$$

where r is some small constant which guarantees that the degree of P(x) is bounded by several thousand (the highest recommended value of r is 13, which for q = 2 gives rise to a polynomial of degree 8192). The bound on the degree is required in order to make it possible to invert P(x) efficiently (e.g., by using Berlekamp's probabilistic algorithm for solving a univariate polynomial equation over a finite fields).

The univariate polynomial P over \mathbf{K} can be expressed as a system of n multivariate polynomials P_1, \ldots, P_{n-1} in the n variables x_0, \ldots, x_{n-1} over \mathbf{F} . The restricted choice of exponents in P guarantees that all the $P_i's$ are homogeneous quadratic polynomials. The trapdoor consists of two random invertible linear transformations S and T over n-tuples of values in \mathbf{F} . The user applies S to the inputs and T to the outputs of the n multivariate polynomials, and publishes the evaluated homogeneous quadratic polynomials in n variables, denoted by G_0, \ldots, G_{n-1} .

To solve the published system of quadratic equations with a given ciphertext as the right hand side, the user applies T^{-1} to the ciphertext, interprets the result as an element of \mathbf{K} , solves his secret univariate polynomial with this right hand side, and applies S^{-1} to the components of the solution. The attacker cannot use this procedure since he does not know the S and T transformations. These mixing operations have natural interpretation over \mathbf{F} but not over \mathbf{K} , and it is not clear a priori that the n published polynomials over \mathbf{F} can be described by a single univariate polynomial G over \mathbf{K} . Even if it exists, it may have an exponential number of coefficients, and even if it is sparse, it may have an exponentially large degree which makes it practically unsolvable.

Remark: In this paper we simplify the original HFE scheme in several inessential ways. In particular, we consider only homogeneous polynomials (the attacker can ignore lower degree monomials), and assume that the representation of **K** over **F** is fixed (by using a different representation, the attacker obtains a different but equally useful version of the secret key).

3 Univariate Representations of Systems of Multivariate Polynomials

The starting point of our attack is the observation that any system of n multivariate polynomials of bounded degree d in n variables over a field \mathbf{F} can be represented as a single sparse univariate polynomial of a special form over an extension field \mathbf{K} of degree n over \mathbf{F} .

We first consider the case of linear multivariate mappings. The mapping $x \to x^q$ is a linear function over \mathbf{K} , and thus any mapping of the form $x \to \sum_{i=0}^{n-1} a_i x^{q^i}$ for fixed coefficients a_0, \ldots, a_{n-1} in \mathbf{K} is also a linear mapping. We need the converse of this result:

Lemma 1. : Let A be a linear mapping from n-tuples to n-tuples of values in \mathbf{F} . Then there are coefficients a_0, \ldots, a_{n-1} in \mathbf{K} such that for any two n tuples over \mathbf{F} , (x_0, \ldots, x_{n-1}) (which represents $x = \sum_{i=0}^{n-1} x_i \omega_i$ in \mathbf{K}) and (y_0, \ldots, y_{n-1}) (which represents $y = \sum_{i=0}^{n-1} y_i \omega_i$ in \mathbf{K}), $(y_0, \ldots, y_{n-1}) = A(x_0, \ldots, x_{n-1})$ if and only if $y = \sum_{i=0}^{n-1} a_i x^{q^i}$.

Proof: There are $q^{(n^2)}$ $n \times n$ matrices over \mathbf{F} and $(q^n)^n$ sums of n monomials over \mathbf{K} , and thus the number of linear mappings and the number of polynomials of this form is identical. Each polynomial represents some linear mapping, and two distinct polynomials cannot represent the same mapping since their difference would be a non zero polynomial of degree q^{n-1} with q^n roots in a field. Consequently, each linear mapping is represented by some univariate polynomial of this type over the extension field. \square

We now generalize this characterization from linear functions to any system of multivariate polynomials:

Lemma 2. Let $P_0(x_0, ..., x_{n-1}), ..., P_{n-1}(x_0, ..., x_{n-1})$ be any set of n multivariate polynomials in n variables over \mathbf{F} . Then there are coefficients $a_0, ..., a_{q^n-1}$ in \mathbf{K} such that for any two n tuples $(x_0, ..., x_{n-1})$ and $(y_0, ..., y_{n-1})$ of elements in \mathbf{F} , $y_j = P_j(x_0, ..., x_{n-1})$ for all $0 \le j \le n-1$ if and only if $y = \sum_{i=0}^{q^n-1} a_i x^i$, where $x = \sum_{i=0}^{n-1} x_i \omega_i$ and $y = \sum_{i=0}^{n-1} y_i \omega_i$ are the elements of \mathbf{K} which correspond to the two vectors over \mathbf{F} .

Proof: The mere existence of the coefficients a_0, \ldots, a_{q^n-1} in **K** is obvious, since any mapping over a finite field can be described by its interpolation polynomial. However, we provide a different proof which enables us to prove in the next lemma the relationship between the degree of the polynomials over **F** and the sparsity of the polynomials over **K**.

Without loss of generality, we can assume that the first basis element is $\omega_0 = 1$. The mapping $(x_0, \ldots, x_{n-1}) \to (x_i, 0, \ldots, 0)$ over \mathbf{F} is linear, and thus has a univariate polynomial representation over \mathbf{K} . To represent the mapping $(x_0, \ldots, x_{n-1}) \to (\prod_{i=0}^{n-1} x_i^{c_i}, 0, \ldots, 0)$, multiply all the univariate polynomials which represent the mappings $(x_0, \ldots, x_{n-1}) \to (x_i, 0, \ldots, 0)$, with their multiplicities c_i (note that this can only be done at the first coordinate, which corresponds to the basis element $\omega_0 = 1$; at at any other coordinate k we would get a

power of ω_k which would spread the resultant monomial all over the vector). By summing the univariate polynomial representations of such monomials with appropriate coefficients we can represent the mapping defined by any multivariate polynomial in the first coordinate of the vector, and zeroes elsewhere. To move the multivariate polynomial to the k-th coordinate of the vector, we multiply all the coefficients of its univariate representation (which are elements of \mathbf{K}) by ω_k . Finally, to represent a system of n (unrelated) multivariate polynomials at the n coordinates of the vector, we construct a representation of each polynomial at the first coordinate, shift it to its proper coordinate and add all the resultant univariate polynomials. \square

An important corollary of this proof is:

Lemma 3. : Let C be any collection of n homogeneous multivariate polynomials of degree d in n variables over F. Then the only powers of x which can occur with non-zero coefficients in its univariate polynomial representation G(x) over K are sums of exactly d (not necessarily distinct) powers of $q: q^{i_1} + q^{i_2} + \ldots + q^{i_d}$. If d is a constant, then G(x) is sparse, and its coefficients can be found in polynomial time.

Proof: Mappings defined by a single variable are linear functions, and thus can be represented as the sum of monomials of the form x^{q^i} , and each monomial contains a single power of q. When we multiply d such polynomials and evaluate the result, we get only powers of x which are the sums of exactly d powers of q. Since G(x) is the sum of such polynomials (multiplied by constants from K), the same is true for G(x).

The degree of G(x) over **K** can be exponentially large, but at most $O(n^d)$ of its coefficients can be non-zero, and for any fixed value of d this is a polynomial number. Once we know that a sparse univariate polynomial representation exists, we can find its coefficients in polynomial time by interpolation based on sufficiently many input/output pairs. \Box

The problem of solving a system of multivariate quadratic equations over a finite field is known to be NP complete. This lemma implies that the problem of solving a single univariate polynomial equation over a finite field is also NP complete. This is a very natural computational problem, and we were thus surprised to discover that its status was not mentioned in any of the standard references on NP completeness. Note that the problem is NP complete when the (sparse) polynomial is represented by the list of its non zero coefficients, but solvable in probabilistic polynomial time by Berlekamp's algorithm if the polynomial is represented by the list of ALL its coefficients.

Consider the published system of quadratic polynomials G_0, \ldots, G_{n-1} in x_0, \ldots, x_{n-1} . Each polynomial can be written as the quadratic form $\overline{x}G_i\overline{x}^t$ where G_i is an $n \times n$ matrix of coefficients \overline{x} , \overline{x} is the row vector of variables

¹ The matrix representation of quadratic forms is not unique, and has to be symmetrized by averaging the matrix and its transpose. In fields of characteristic 2 we just add the matrix and its transpose, (since we cannot divide by 2), and use the result. More details on these fine points will be given in the final version of the paper.

 (x_0, \ldots, x_{n-1}) , and \overline{x}^t is its transpose. However, our attack does not use this standard representation. Instead, it uses Lemma 3.3 to efficiently find the following representation of the public key:

$$G(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} g_{ij} x^{q^i + q^j} = \underline{x} G \underline{x}^t \text{ where } G = [g_{ij}] \text{ and } \underline{x} = (x^{q^0}, x^{q^1}, \dots, x^{q^{n-1}})$$

Note that this is an unusual type of quadratic form since the vector \underline{x} consists of related rather than independent variables, and that \overline{x} is a vector of elements from \mathbf{F} whereas \underline{x} is a vector of elements from \mathbf{K} . It is this representation which makes it possible to analyze the secret hiding operations in a clean mathematical form.

4 The Effect of S and T on P

Due to their special form, both the original polynomial P(x) over \mathbf{K} chosen by the user and the new polynomial G(x) over \mathbf{K} derived by the cryptanalyst from the public key can be represented by the (non standard) quadratic forms $\underline{x}P\underline{x}^t$ and $\underline{x}G\underline{x}^t$. The linear mappings S and T can be represented as univariate polynomials, and thus the public key is represented by the univariate polynomial composition G(x) = T(P(S(x))) over \mathbf{K} . We rewrite this equation as $T^{-1}(G(x)) = P(S(x))$, where S has the form $S(x) = \sum_{i=0}^{n-1} s_i x^{q^i}$ and T^{-1} (which is also a linear mapping) has the form $T^{-1}(x) = \sum_{i=0}^{n-1} t_i x^{q^i}$. Our goal now is to study the effect of the polynomial compositions $T^{-1}(G(x))$ and P(S(x)) on the matrices of their (non standard) quadratic form representations.

Theorem 4.: The matrix of the quadratic form in \underline{x} which represents the polynomial composition $T^{-1}(G(x))$ is $\sum_{k=0}^{n-1} t_k G^{*k}$ where G^{*k} is obtained from the $n \times n$ matrix representation of G by raising each one of its entries to the power q^k in \mathbf{K} , and cyclically rotating forwards by k steps both the rows and the columns of the result. The matrix of the quadratic form in \underline{x} which represents the polynomial composition P(S(x)) is WPW^t in which $W = [w_{ij}]$ is an $n \times n$ matrix defined by $w_{ij} = (s_{j-i})^{q^i}$, where j-i is computed modulo n.

Proof (Sketch): The polynomial representation of $T^{-1}(x)$ is $\sum_{k=0}^{n-1} t_k x^{q^k}$ and the polynomial representation of G(x) is $\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} g_{ij} x^{q^i+q^j}$. Their polynomial composition can be evaluated by using the fact that raising sums to the power q^i is a linear operation:

$$T^{-1}(G(x)) = \sum_{k=0}^{n-1} t_k \left(\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} g_{ij} x^{q^i + q^j}\right)^{q^k} = \sum_{k=0}^{n-1} t_k \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (g_{ij})^{q^k} x^{(q^i + q^j)q^k}$$

The exponents of q can be reduced modulo n since $x^{q^n} = x^{q^0} = x$, and the summation indices can be cyclically rotated if they are computed modulo n:

$$T^{-1}(G(x)) = \sum_{k=0}^{n-1} t_k \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (g_{ij})^{q^k} x^{q^{i+k} + q^{j+k}} = \sum_{k=0}^{n-1} t_k \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (g_{i-k,j-k})^{q^k} x^{q^i + q^j}$$

The matrix of the quadratic form representation of this polynomial in terms of $\underline{\mathbf{x}}$ is exactly $G' = \sum_{k=0}^{n-1} t_k G^{*k}$, where the (i,j)-th entry of G^{*k} is $g_{i-k,j-k}^{q^k}$, as specified.

The proof of the other type of composition is similar:

$$P(S(x)) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} p_{ij} (\sum_{k=0}^{n-1} s_k x^{q^k})^{(q^i + q^j)} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} p_{ij} (\sum_{u=0}^{n-1} s_u x^{q^u})^{q^i}) (\sum_{v=0}^{n-1} s_v x^{q^v})^{q^j})$$

Again we use linearity and cyclic index shifting to evaluate P(S(x)) as:

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}p_{ij}(\sum_{u=0}^{n-1}s_u^{q^i}x^{q^{u+i}})(\sum_{v=0}^{n-1}s_v^{q^j}x^{q^{v+j}}) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}p_{ij}(\sum_{u=0}^{n-1}s_{u-i}^{q^i}x^{q^u})(\sum_{v=0}^{n-1}s_{v-j}^{q^j}x^{q^v})$$

By rearranging the order of the summation and the multiplied terms we get:

$$P(S(x)) = \sum_{n=0}^{n-1} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{n=0}^{n-1} x^{q^u} s_{u-i}^{q^i} p_{ij} s_{v-j}^{q^j} x^{q^v} = \underline{x} W P W^t \underline{x}^t$$

where W is the specified matrix. \Box

5 Recovering the Secret Key from the Public Key

The attack on a given public key is based on the matrix equation over K, $G' = WPW^t$, which we call the fundamental equation. The matrix G can be easily computed by representing the public key as a univariate polynomial over K, and then representing the univariate polynomial as the quadratic form $\underline{x}G\underline{x}^t$. All the G^{*k} variants of G can be computed by raising the entries of G to various powers and cyclically rotating its rows and columns. We can thus consider $G' = \sum_{k=0}^{n-1} t_k G^{*k}$ as a matrix whose entries are linear combinations of known values with unknown coefficients t_0, \ldots, t_{n-1} from K. The matrix P is mostly known, since only the top left $r \times r$ block in the $n \times n$ matrix can be non zero, and r << n. The matrix W is unknown, but there are many relations between its n^2 entries since they are all determined by just n parameters via $w_{ij} = s_{j-i}^{q^i}$. Our goal is to use all these observations in order to solve the fundamental equation in polynomial time.

5.1 Recovering T

We first describe the process of recovering t_0, \ldots, t_{n-1} from the fundamental equation $G' = WPW^t$, where each entry in G' is a linear combination of the t_k variables. The matrix P contains at most r non zero rows, and thus both its rank and the rank of WPW^t cannot exceed r. For random choices of t_k values the expected rank of the evaluated G' matrix is close to n. What makes the correct choice of t_k values special is that they force G' to have the unusually small rank

r. To simplify the asymptotic analysis, we assume that r is a constant and n grows to infinity, and argue that the attack should run in expected polynomial time (even though we cannot formally prove this claim).

The basic approach is to express this rank condition as a large number of equations in a small number of variables. Consider the matrix G' evaluated with the correct choice of t_k values. Its rank is at most r, and thus its left kernel (defined as the set of all row vectors \widetilde{x} over \mathbf{K} satisfying $\widetilde{x}G'=0$) is a n-r dimensional linear subspace. We thus expect to find in it n-r linearly independent vectors $\widetilde{x}_1,\ldots,\widetilde{x}_{n-r}$ even if we force the first n-r entries in each \widetilde{x}_k to have some arbitrarily specified values. The remaining r entries in each one of the n-r vectors \widetilde{x}_k are defined as new variables. Each vector equation $\widetilde{x}G'=0$ can be viewed as n scalar equations over \mathbf{K} , and thus we get a total of n(n-r) equations in the r(n-r)+n variables (the original t_k coefficients in G' and the new unspecified entries in all the \widetilde{x}_k vectors).

The bad news is that the equations are quadratic, and we don't know how to solve large systems of quadratic equations in polynomial time (in fact, this was the original problem of deriving cleartexts from ciphertexts!). The good news is that instead of a marginally defined system of n equations in n variables, we get an overdefined system of about n^2 equations in about n variables where n < n.

Consider the general problem of solving e randomly generated homogeneous quadratic equations in m variables. The well known linearization technique for solving such equations is to replace any product of two variables x_ix_j for $i \leq j$ by a new variable y_{ij} . The total number of new variables is n(n+1)/2. Each quadratic equation in the original x variables can be rewritten as a linear equation in the new y variables. If the number of equations satisfies $e \geq n(n+1)/2$, we expect the system to be uniquely solvable, but if e << n(n+1)/2, we expect the linear system to have an exponential number of parasitic y solutions which do not correspond to any real x solution.

Unfortunately, in our problem we have $m \approx rn$ variables but only $e \approx \epsilon m^2$ quadratic equations where $\epsilon = 1/r^2$ is smaller than 1/2, and thus the linearization method would fail. In the next subsection we describe a novel heuristic technique called *relinearization* which is expected to solve such systems of quadratic equations for any fixed $\epsilon > 0$ in polynomial time. The technique seems to have many other applications in cryptography, optimization, and computer algebra, and should be studied carefully.

5.2 The Relinearization Technique

Consider a system of ϵm^2 homogeneous quadratic equations in the m variables x_1, \ldots, x_m . We rewrite it as a new system of ϵm^2 linear equations in the (approximately) $m^2/2$ new variables $y_{ij} = x_i x_j$ for $i \leq j$. Its solution space is a linear subspace of expected dimension $(1/2 - \epsilon)m^2$, and each solution can be expressed as a linear function of $(1/2 - \epsilon)m^2$ new variables z_k . Such a parametric solution can be efficiently found by Gauss elimination.

Most of the y_{ij} solutions found in this way do not correspond to any possible x_i solutions. We want to add additional constraints which relate the various y_{ij} variables to each other in the way implied by their definition as $y_{ij} = x_i x_j$. To do this, consider any 4-tuple of indices $1 \le a \le b \le c \le d \le m$. Then $x_a x_b x_c x_d$ can be parenthesized in three different ways:

$$(x_a x_b)(x_c x_d) = (x_a x_c)(x_b x_d) = (x_a x_d)(x_b x_c) \implies y_{ab} y_{cd} = y_{ac} y_{bd} = y_{ad} y_{bc}$$

There are about $m^4/4!$ different ways to choose sorted 4-tuples of distinct indices, and each choice gives rise to 2 equations 2 . We thus get about $m^4/12$ quadratic equations in the $m^2/2$ y_{ij} variables, and it is not difficult to prove that they are linearly independent (even though they are algebraically dependent). We can lower the number of variables to $(1/2 - \epsilon)m^2$ by replacing each one of the y_{ij} variables by its parametric representation as a linear combination of the new z_k variables.

The relinearization technique is based on the observation that the new $m^4/12$ quadratic equations in the new $(1/2 - \epsilon)m^2$ z_i variables can be linearized again by replacing each product $z_i z_j$ for $i \leq j$ by a new variable v_{ij} . The new system has $m^4/12$ linear equations in $((1/2 - \epsilon)m^2)^2/2$ v_{ij} variables. We expect this linear system to be uniquely solvable when $m^4/12 \geq ((1/2 - \epsilon)m^2)^2/2$. This is satisfied whenever $\epsilon \geq 1/2 - 1/\sqrt{6} \approx 0.1$, which is one fifth of the number of equations required by simple linearization.

Two small demonstrations of this procedure can be found in the appendix. There are many possible optimizations of the basic technique: we can use relinearization recursively, consider additional constraints, etc. For example, there are about $m^6/6!$ possible choices of indices in $x_a x_b x_c x_d x_e x_f$, and each one gives rise to 14 different equations of degree 3 in the $(1/2 - \epsilon)m^2$ parameters z_i . If we relinearize every product of the form $z_i z_j z_k$ for $i \le j \le k$, we get about $14m^6/720$ linear equations in $((1/2 - \epsilon)m^2)^3/6$ new variables, which can be solved whenever $\epsilon \ge 0.008$. In the full version of this paper we show that for any fixed $\epsilon > 0$ there is a relinearization scheme which is expected to solve in polynomial time random systems of ϵm^2 quadratic equations in m variables.

We now return to the original problem of extracting T from the fundamental equation $G' = WPW^t$. Since we have about n^2 quadratic equations in about rn variables, we get $\epsilon \approx 1/r^2$. The worst case happens when q=2 and r=13, yielding $\epsilon \approx 0.006$, which is marginally smaller than the threshold stated above. We thus have to use the relinearization scheme which considers products of 8 x_i values, and to solve a huge system of $O(n^8)$ linear equations in $O(n^8)$ variables, which is polynomial but impractical. For larger fields \mathbf{F} , both r and n drop considerably if we keep fixed both the degree q^r of the secret polynomial and the size q^n of the cleartext space (for example, when we replace $\mathbf{F_2}$ by $\mathbf{F_7}$, r drops from 13 to 4 and n drops from 100 to 36). The smaller r increases ϵ and makes it possible to use simpler relinearization schemes which result in smaller systems of $O(n^6)$ or even $O(n^4)$ equations, and the smaller n make their solution more

² There are additional 4-tuples of non-distinct indices, which give either one or no additional equations. We ignore them in our asymptotic analysis.

feasible. The practical details are messy, and will be omitted from this extended abstract.

One final complication is the fact that the quadratic equations have multiple solutions (due to two symmetries of the fundamental equation: we can raise all the t_i to the power q and cyclically rotate the vector to the right, and we can multiply all of them by a common constant). Any linearized technique to find these solutions will necessarily return the n dimensional linear subspace they span. Almost all the points on this subspace are parasitic solutions, which do not solve the original quadratic equations, and cannot be used to break the scheme. To avoid this problem, we want to force the system of quadratic equations to have a unique solution. The standard way to do this is to choose random additional constraints until only one of the original solutions remains. However, the equations are over the large field \mathbf{K} , and each additional equation kills all but $1/q^n \approx 2^{-100}$ of the original solutions, which is too severe. Instead, we can reexpress the quadratic equations over the large field ${\bf K}$ as quadratic equations over the small field **F**, and arbitrarily fix the values of some of the new variables in F. Each additional choice reduces the number of solutions by the small factor q, and with reasonable probability the number of solutions will pass through 1. Working over **F** instead of **K** increases the number of variables by another factor of n, but we can avoid this higher complexity by translating to **F** only the O(n) parameters of the linear solution space rather than the $O(n^8)$ variables of the linearized problem. We can then express some of the algebraic relationships between the $O(n^8)$ linearized variables as quadratic equations in the new $O(n^2)$ variables over F. The number of quadratic equations we get exceeds the square of the number of new variables, and thus we can solve them efficiently by simple linearization.

5.3 Recovering S

The last part of the attack recovers S and P when T is known. The matrix $G' = \sum_{k=0}^{n-1} t_k G^{*k}$ in the fundamental equation $G' = WPW^t$ is now a completely known matrix. The matrix P contains at most r non zero rows, and thus both its rank and the rank of $G' = WPW^t$ cannot exceed r. Assume without loss of generality that the rank of P is exactly r, and that the rank of W is exactly r. Let v_1, \ldots, v_{n-r} be a basis for the left kernel of G'. Since W^t is invertible the left kernel of WPW^t is equal to the left kernel of WP. The left kernel of P consists of exactly those vectors which are zero in their first r entries, and thus each v_i is mapped by W to a vector of this form. Since G' is known, its left kernel can be easily computed, and each one of the n-r basis vectors gives rise to r equations in the unknown entries of W.

The problem seems to be underdefined, with r(n-r) linear equations in n^2 variables. We can reduce the number of variables from n^2 to n by replacing each w_{ij} by s_{j-i}^q , but then we get nonlinear equations. The crucial observation is that these nonlinear equations over \mathbf{K} become linear if we reinterpret them as equations over \mathbf{F} : Replace each s_i by $\sum_{j=1}^{n-1} s_{ij}\omega_j$ where the s_{ij} is a new set of n^2

variables over \mathbf{F} . Each $s_{j-i}^{q^i}$ becomes a linear combination of the s_{uv} variables, and each equation over \mathbf{K} becomes a collection of n linear equations over \mathbf{F} . Altogether there are r(n-r)n equations in the n^2 new variables over \mathbf{F} , and for any r>1 the system is greatly overdefined since $r(n-r)n>>n^2$. The solution of the homogeneous equations can be defined at most up to multiplication by a constant, but as explained earlier any solution of this type is satisfactory.

A Appendix: A Relinearization Example

We demonstrate the complete relinearization technique on a toy example of 5 random quadratic equations in three variables x_1, x_2, x_3 modulo 7:

$$3x_1x_1 + 5x_1x_2 + 5x_1x_3 + 2x_2x_2 + 6x_2x_3 + 4x_3x_3 = 5$$

$$6x_1x_1 + 1x_1x_2 + 4x_1x_3 + 4x_2x_2 + 5x_2x_3 + 1x_3x_3 = 6$$

$$5x_1x_1 + 2x_1x_2 + 6x_1x_3 + 2x_2x_2 + 3x_2x_3 + 2x_3x_3 = 5$$

$$2x_1x_1 + 0x_1x_2 + 1x_1x_3 + 6x_2x_2 + 5x_2x_3 + 5x_3x_3 = 0$$

$$4x_1x_1 + 6x_1x_2 + 2x_1x_3 + 5x_2x_2 + 1x_2x_3 + 4x_3x_3 = 0$$

After replacing each $x_i x_j$ by y_{ij} , we solve the system of 5 equations in 6 variables to obtain a parametric solution in a single variable z:

$$y_{11} = 2 + 5z$$
, $y_{12} = z$, $y_{13} = 3 + 2z$, $y_{22} = 6 + 4z$, $y_{23} = 6 + z$, $y_{33} = 5 + 3z$

This single parameter family contains 7 possible solutions, but only two of them also solve the original quadratic system. To filter out the parasitic solutions, we impose the additional constraints: $y_{11}y_{23} = y_{12}y_{13}$, $y_{12}y_{23} = y_{13}y_{22}$, $y_{12}y_{33} = y_{13}y_{23}$. Substituting the parametric expression for each y_{ij} , we get:

$$(2+5z)(6+z) = z(3+2z), \ z(6+z) = (3+2z)(6+4z), \ z(5+3z) = (3+2z)(6+z)$$

These equations can be simplified to:

$$3z^2 + z + 5 = 0$$
, $0z^2 + 4z + 4 = 0$, $1z^2 + 4z + 3 = 0$

The relinearization step introduces two new variables $z_1 = z$ and $z_2 = z^2$, and treats them as unrelated variables. We have three linear equations in these two new variables, and their unique solution is $z_1 = 6$, $z_2 = 1$. Working backwards we find that $y_{11} = 4$, $y_{22} = 2$, $y_{33} = 2$, and by extracting their square roots modulo 7 we find that $x_1 = \pm 2$, $x_2 = \pm 3$, $x_3 = \pm 3$. Finally, we use the values $y_{12} = 6$ and $y_{23} = 5$ to combine these roots in just two possible ways to obtain $x_1 = 2$, $x_2 = 3$, $x_3 = 4$ and $x_1 = 5$, $x_2 = 4$, $x_3 = 3$, which solve the original quadratic system.

A larger example of a solvable system consists of 5 randomly generated homogeneous quadratic equations in 4 variables, Note that this is barely larger than the minimum number of equations required to make the solution well defined.

The number of linearized variables $y_{ij} = x_i x_j$ for $1 \le i \le j \le 4$ is 10, and the solution of the system of 5 linear equations in these 10 variables can be defined by affine expressions in 5 new parameters z_i . There are 20 equations which can be derived from fundamentally different ways of parenthesizing products of $4 x_i$ variables:

```
y_{12}y_{34} = y_{13}y_{24} = y_{14}y_{23}
y_{11}y_{23} = y_{12}y_{13}, \quad y_{11}y_{24} = y_{12}y_{14}, \quad y_{11}y_{34} = y_{13}y_{14}
y_{22}y_{13} = y_{12}y_{23}, \quad y_{22}y_{14} = y_{12}y_{24}, \quad y_{22}y_{34} = y_{23}y_{24}
y_{33}y_{12} = y_{13}y_{23}, \quad y_{33}y_{14} = y_{13}y_{34}, \quad y_{33}y_{24} = y_{23}y_{34}
y_{44}y_{12} = y_{14}y_{24}, \quad y_{44}y_{13} = y_{14}y_{34}, \quad y_{44}y_{23} = y_{24}y_{34}
y_{11}y_{22} = y_{12}y_{12}, \quad y_{11}y_{33} = y_{13}y_{13}, \quad y_{11}y_{44} = y_{14}y_{14}
y_{22}y_{33} = y_{23}y_{23}, \quad y_{22}y_{44} = y_{24}y_{24}, \quad y_{33}y_{44} = y_{34}y_{34}
```

When we substitute the affine expressions in the 5 new z_i parameters and relinearize it, we get 20 linear equations in the 5 z_i and their 15 products $z_i z_j$ for $1 \le i \le j \le 5$, which is just big enough to make the solution unique (up to \pm sign) with reasonable probability. \Box

References

- CSV97. D. Coppersmith, J. Stern and S. Vaudenay, *The Security of the Birational Permutation Signature Scheme*, Journal of Cryptology, 1997, pp. 207-221.
- FD85. H. Fell and W. Diffie, Analysis of a Public Key Approach Based on Polynomial Substitution, Crypto 85, Springer Verlag, pp. 340-349.
- KS98. A. Kipnis and A. Shamir, Cryptanalysis of the Oil and Vinegar Signature Scheme, Crypto 98, Springer Verlag, pp. 257-266.
- K98. N. Koblitz Algebraic Aspects of Cryptography, Springer Verlag, 1998.
- MI88. T. Matsumoto and H. Imai, Public Quadratic Polynomial Tuples for Efficient Signature Verification and Message Encryption, Eurocrypt 88, Springer Verlag, pp. 419-453.
- OSS84. H. Ong, C. P. Schnorr, and A. Shamir A Fast Signature Scheme Based on Quadratic Equations, Proc. 16-th ACM Symp. Theory of Computation, 1984, pp. 208-216.
- P95. J.Patarin, Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt 88, Crypto 95, Springer Verlag, pp.248-261.
- P96a. J.Patarin, Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms, Eurocrypt 96, Springer Verlag, pp.33-48.
- P96b. J. Patarin, Asymmetric Cryptography with a Hidden Monomial, Crypto 96, Springer Verlag, pp. 45-60.
- P97. J. Patarin, The Oil and Vinegar Algorithm for Signatures, presented at the Dagstuhl Workshop on Cryptography, September 97.
- PS87. J. M. Pollard and C. P. Schnorr, An Efficient Solution of the Congruence $x^2 + ky^2 = m \pmod{n}$, IEEE Trans. Information Theory, vol. IT-33, no. 5, 1987, pp. 702-709.
- S93. A. Shamir Efficient Signature Schemes Based on Birational Permutations, Crypto 93, Springer Verlag, pp.1-12.

The Hardness of the Hidden Subset Sum Problem and Its Cryptographic Implications

Phong Nguyen and Jacques Stern

École Normale Supérieure
Laboratoire d'Informatique
45 rue d'Ulm, 75230 Paris Cedex 05
France
{Phong.Nguyen, Jacques.Stern}@ens.fr
http://www.dmi.ens.fr/~{pnguyen,stern}/

Abstract. At Eurocrypt'98, Boyko, Peinado and Venkatesan presented simple and very fast methods for generating randomly distributed pairs of the form $(x, q^x \mod p)$ using precomputation. The security of these methods relied on the potential hardness of a new problem, the so-called hidden subset sum problem. Surprisingly, apart from exhaustive search, no algorithm to solve this problem was known. In this paper, we exhibit a security criterion for the hidden subset sum problem, and discuss its implications on the practicability of the precomputation schemes. Our results are twofold. On the one hand, we present an efficient lattice-based attack which is expected to succeed if and only if the parameters satisfy a particular condition that we make explicit. Experiments have validated the theoretical analysis, and show the limitations of the precomputation methods. For instance, any realistic smart-card implementation of Schnorr's identification scheme using these precomputations methods is either vulnerable to the attack, or less efficient than with traditional precomputation methods. On the other hand, we show that, when another condition is satisfied, the pseudo-random generator based on the hidden subset sum problem is strong in some precise sense which includes attacks via lattice reduction. Namely, using the discrete Fourier transform, we prove that the distribution of the generator's output is indistinguishable from the uniform distribution. The two conditions complement each other quite well, and therefore form a convincing picture of the security level.

1 Introduction

In many discrete-log-based protocols, one needs to generate pairs of the form $(x, g^x \mod p)$ where x is random and g is a fixed base. ElGamal [9] and DSS [13] signatures, Schnorr's [18,19] and Brickell-McCurley's [4] schemes for identification and signature are examples of such protocols. The generation of these pairs is often the most expensive operation, which makes it tempting to reduce the number of modular multiplications required per generation, especially for smartcards. There are basically two ways to solve this problem. One way is to generate separately a random x, and then compute $g^x \mod p$ using a precomputation

method [3,7,12]. The other way is to generate x and $g^x \mod p$ together by a special pseudo-random number generator which uses precomputations. Schnorr was the first to propose such a preprocessing scheme [18]. The scheme had much better performances than all other methods but there was a drawback: the ouptut exponent x was no more guaranteed to be random, and therefore, each generation might leak information. Indeed, de Rooij [6] showed how to break the scheme. Schnorr later proposed a modified version [19], which was also broken by de Rooij [8].

At Eurocrypt'98, Boyko, Peinado and Venkatesan proposed new and very simple generators [2] to produce pairs of the form $(x, g^x \mod p)$, which could reduce even further the number of necessary modular multiplications. The security of these methods apparently depended on a new problem, the so-called hidden subset sum problem: given a positive integer M and $b_1, \ldots, b_m \in \mathbb{Z}_M$, find $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_M$ such that each b_i is some subset sum modulo M of $\alpha_1, \ldots, \alpha_n$. The problem borrows its name from the classical subset sum problem: given a positive integer M and $b, \alpha_1, \ldots, \alpha_n \in \mathbb{Z}_M$, find $S \subset \{1, \ldots, n\}$ such that $b \equiv \sum_{i \in S} \alpha_i \pmod{M}$. The most powerful known attack [5] against the subset sum problem reduces it to a shortest vector problem in a lattice built from $b, \alpha_1, \ldots, \alpha_n, M$. Provided a shortest vector oracle, the method succeeds with high probability if the density, defined as $d = n/\log_2 M$, is small, namely less than a constant approximately equal to 0.94. However, this method can hardly be applied to hidden subset sums: one cannot even build the lattice since the α_i 's are hidden. Actually, apart from exhaustive search, no algorithm was known to solve the hidden subset sum problem. And thus, according to the authors of [2], the problem was potentially harder than the subset sum problem. Still, they suggested high values of parameters to prevent any subset sum attack, for unknown reasons. For these choices of parameters, the scheme was not suited for smartcards, and the speed-up over other methods was questionable.

It was therefore natural to ask whether or not, one could select small parameters in order to make the scheme very efficient, without affecting the security. More generally, Boyko et al. raised the following question: how hard is the hidden subset sum problem? The present paper provides an answer. We exhibit a security criterion for the hidden subset sum problem which is twofold. On the one hand, we present an efficient lattice-based algorithm to solve the hidden subset sum problem. It relies on a systematic use of the powerful notion of an orthogonal lattice, which was introduced at Crypto'97 [14] by Nguyen and Stern as a cryptographic tool, and subsequently used in cryptanalysis [16,15]. The algorithm is very different from known lattice-based methods to solve subset sums, but surprisingly, seems to generalize their results. More precisely, our algorithm is expected to succeed when the density $d = n/\log_2 M$ is very small. Unfortunately, this is exactly the case arising when one wants to make the scheme practical and more efficient than other exponentiation methods, in a smart-card environment. We have implemented the algorithm, and experiments have confirmed our analysis. On the other hand, we show that when the density is high, the pseudo-random generator based on the hidden subset sum problem is strong

in some precise sense. Namely, using the discrete Fourier transform, we prove that the distribution of the generator's output is then statistically close to the uniform distribution. Such a result was already known (related results in [1,10]), but our proof technique is different. Those results tend to prove that the hardness of the hidden subset sum problem is measured by the density, as for the subset sum problem.

The remainder of the paper is organized as follows. In section 2, we describe the generators of pairs $(x, g^x \mod p)$ proposed at Eurocrypt'98 in [2], and we clarify the relationships between the security of these schemes and the hidden subset sum problem. In section 3, we recall some facts on orthogonal lattices from [14]. Section 4 presents our lattice-based algorithm to solve hidden subset sum problems, and the experiments. In section 5, we discuss the hardness of the hidden subset problem, by measuring the randomness of the generator output.

2 Fast Exponentiation with Hidden Subset Sums

Let p be a prime number, and $g \in \mathbb{Z}_p^*$ of order M. In [2], several generators producing pairs $(x, g^x \mod p)$ were proposed. The simplest generator was the following one:

Preprocessing Step: Generate n random integers $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_M$. Compute $\beta_j = g_j^{\alpha}$ for each j and store both α_j 's and β_j 's in a table.

Pair Generation: Whenever a pair (x, g^x) is needed, randomly generate $S \subseteq \{1, \ldots, n\}$ such that $|S| = \kappa$. Compute $b = \sum_{j \in S} \alpha_j \mod M$. If b = 0, stop and start again. Compute $B = \prod_{j \in S} \beta_j \mod p$ and return (b, B).

Clearly, for any output (b, B), we have $B = g^b \mod p$. The other generators are just variants of the previous generator, using random walks. We will not discuss those, since the security of the generators relies on the same problem.

2.1 Parameters

The scheme needs to store n elements of \mathbb{Z}_M , and n elements of \mathbb{Z}_p^* . Recall that for DSS [13] and Schnorr [18,19], M has 160 bits, while for ElGamal [9] and Brickell-McCurley [4], M has at least 512 bits. Each generation requires κ modular multiplications. When $\kappa \ll n/2$, we say that the underlying hidden subset sum problem is sparse. The parameters n and κ must be sufficiently large to prevent from birthday attacks. In [2], it was suggested to choose n=512 and $\kappa=64$. Comparisons with traditional precomputation methods were made, but only in the case of 512-bit exponents. Table 1 compares the scheme with several configurations of the simple exponentiation method with precomputation of [12]. It shows that for a 160-bit exponent, the generator with the proposed parameters is worse in all aspects. For a 512-bit exponent, it is better: with similar storage, one gains 14 multiplications. But with other precomputation methods, there is no security issue since the exponent is random. Another issue is the viability of the scheme for low-computing-power devices. For instance, a storage of 672 represents 42 Kbytes, which is unacceptable for a smartcard.

Table 1. A comparison of methods for generating pairs $(x, g^x \mod p)$ where p is a 512-bit prime. Storage requirements are in 512-bit numbers. Times are in multiplications per exponentiation.

	160-bit exponent		512-bit exponent	
Method	Storage	Time	Storage	Time
Hidden subset sum generator	672	64	1024	64
Lim and Lee [12]	30	58	62	153
	62	46	157	106
	508	27	1020	78

Thus, the parameters proposed in [2] are rather suited for server applications. In order to offer much better performances than other methods, one is tempted to decrease the parameters. We will discuss possible parameters when we present the experiments related to our attack.

2.2 Security Against Active Attacks

When the generator is used, the security seems to rely on the underlying hidden subset sum problem. Indeed, suppose for instance that the generator is used in Schnorr's [19] identification scheme. Let q be a 160-bit prime dividing p-1, where p is a 512-bit prime.

The prover has a secret key $s \in \mathbb{Z}_q^*$ and a public key $v = g^{-s} \mod p$, where g is a primitive qth root of unity. He generates a random pair $(k, g^k \mod p)$ and sends $x = g^k$ to the verifier. The verifier returns a challenge $e \in \mathbb{Z}_q$. Then the prover sends $y = k + es \mod q$. Finally, the verifier checks whether $x = g^y v^e \mod p$. In an active attack, the verifier can issue many times the challenge $0 \in \mathbb{Z}_q$. He thus gets many outputs of the generator, as y = k. After solving the underlying hidden subset sum problem, he knows the hidden $\alpha_1, \ldots, \alpha_n$. He then issues the challenge $1 \in \mathbb{Z}_q$, to obtain $k + s \mod q$ for some unknown k a subset sum of the α_j 's. If n and κ are not too large, he can exhaustively search for the 0,1-coefficients of the α_j 's to disclose k, and hence the secret key s.

Conversely, if the output of the hidden subset sum generator used is cryptographically pseudo-random, then the speeded-up versions of the following schemes are secure against polynomial time adaptive attacks, provided that the original schemes are secure: ElGamal, DSS and Schnorr signatures, Schnorr identification. (see [2]).

2.3 Security Against Passive Attacks

In [2] (Theorems 6 and 7, p.230), it was claimed that only the security against active attacks needed to assume the hardness of the hidden subset sum problem. However, it seems that the security against passive attacks actually relies on the potential hardness of a slight variant of the hidden subset sum problem,

which we call the affine hidden subset sum problem: given a positive integer M, and $b_1, \ldots, b_m, c_1, \ldots, c_m \in \mathbb{Z}_M$, find integers $s, \alpha_1, \ldots, \alpha_n \in \mathbb{Z}_M$ such that each $b_i + sc_i$ is some subset sum modulo M of $\alpha_1, \ldots, \alpha_n$.

Assume for instance that the generator is used in Schnorr's signature scheme. We keep the notations of the previous section. The public key is $v = g^{-s} \mod p$. The signer generates a random pair $(k, g^k \mod p)$. He computes a hash e = h(k, m) where m is the message, and $y = k + es \mod q$. The signature is the pair (y, e). Notice that $k = y - es \mod q$ is a hidden subset sum, where y and e are known and e is secret. Thus, a passive attacker is left with an affine hidden subset sum problem with the pairs (y, -e) and the modulus e. If he can solve this problem, he recovers the secret key e.

The previous remark can be adapted to the following schemes: Schnorr's and Brickell-McCurley's identification, ElGamal and DSS signatures. For example, in the case of DSS, a signature is of the form (a,b) where $b=k^{-1}(m+as) \mod q$, s is the secret key and m is the hash. Note that $k=mb^{-1}+ab^{-1}s \mod q$ is a hidden subset sum. But mb^{-1} and ab^{-1} are known, so this is again an affine hidden subset sum problem, from which one can derive the secret key s.

We will see that our attack against the hidden subset sum problem can be adapted to the affine hidden subset sum problem. It appears that the complexity of these problems is similar.

3 Lattice Reduction and the Orthogonal Lattice

Throughout the paper, we call lattice any subgroup of \mathbb{Z}^m for some integer m. If L is a lattice, we denote by $\det(L)$ its determinant (or volume), and $\Lambda(L)$ the Euclidean norm of a shortest non-zero vector of L. A classical result of Minkowski states that for any integer d, there is a constant $\gamma(d)$ such that for all d-dimensional lattice L:

$$\Lambda(L) \le \gamma(d) \det(L)^{1/d}$$
.

The smallest such constant is denoted by γ_d and called Hermite's constant of rank d. It is known that:

$$\sqrt{\frac{d}{2\pi e}} \le \gamma_d \le \sqrt{\frac{d}{\pi e}}.$$

As a result, it is convenient to assume that for a "random" d-dimensional lattice L, the quantity $\Lambda(L)/(\sqrt{d}\det(L)^{1/d})$ is roughly equal to some universal constant γ . The goal of lattice reduction is to find a reduced basis, that is, a basis consisting of reasonably short vectors. In the sequel, we will not need more precise definitions, or very precise approximations for the shortest vector. In practice, one hopes to obtain sufficiently reduced bases thanks to reduced bases in the sense of LLL [11], or its variants [17,20].

Let L be a lattice in \mathbb{Z}^m . The orthogonal lattice L^{\perp} is defined as the set of elements in \mathbb{Z}^m which are orthogonal to all the lattice points of L, with respect to the usual dot product. We define the lattice $\bar{L} = (L^{\perp})^{\perp}$, which is the intersection

of \mathbb{Z}^m with the \mathbb{Q} -vector space generated by L: it contains L and its determinant divides the one of L. The result of [14] which are of interest to us is the following one:

Theorem 1. If L is a lattice in \mathbb{Z}^m , then $\dim(L) + \dim(L^{\perp}) = m$ and $\det(L^{\perp})$ is equal to $\det(\bar{L})$.

This suggests that if L is a "random" low-dimensional lattice in \mathbb{Z}^m , a reduced basis of L^{\perp} will consist of very short vectors compared to a reduced basis of L. More precisely, one expects that any reduced basis of L^{\perp} will consist of vectors with norm around $\gamma \sqrt{m - \dim L} \det(\bar{L})^{1/(m - \dim L)}$. Furthermore, one can note that computing a basis of the orthogonal lattice amounts to compute the integer kernel of an (integer) matrix, so that:

Theorem 2. There exists an algorithm which, given as input a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a lattice L in \mathbb{Z}^m , outputs a basis of the orthogonal lattice L^{\perp} , and whose running time is polynomial with respect to m, d and any upper bound of the bit-length of the $\|\mathbf{b}_j\|$'s.

In fact, it was proved in [14] that one could directly obtain an LLL-reduced basis of the orthogonal lattice by a suitable LLL-reduction, in polynomial time.

4 A Lattice-Based Attack

Let us first restate the hidden subset sum problem in terms of vectors. Given an integer M, and a vector $\mathbf{b} = (b_1, \dots, b_m) \in \mathbb{Z}^m$ with entries in [0..M-1], find integers $\alpha_1, \dots, \alpha_n \in [0..M-1]$ such that there exist vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{Z}^m$ with entries in $\{0,1\}$ satisfying:

$$\mathbf{b} = \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \dots + \alpha_n \mathbf{x}_n \pmod{M}$$
 (1)

Throughout this section, we will assume that (\mathbf{b}, M) is a correct input. That is, there exist integers $\alpha_1, \ldots, \alpha_n \in [0..M-1]$, vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{Z}^m$ with entries in $\{0, 1\}$, and a vector $\mathbf{k} \in \mathbb{Z}^m$ such that:

$$\mathbf{b} = \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \dots + \alpha_n \mathbf{x}_n + M \mathbf{k}$$
 (2)

Our attack proceeds in three steps:

- 1. From **b**, we determine the lattice $\bar{L}_{\mathbf{x}}$, where $L_{\mathbf{x}}$ is the lattice generated by the \mathbf{x}_{j} 's and \mathbf{k} .
- 2. From $\bar{L}_{\mathbf{x}}$, we derive the hidden coefficients \mathbf{x}_j 's.
- 3. Using **b**, the \mathbf{x}_j 's and M, we finally recover the weights α_j 's.

Note that this attack recovers all secret data, not just the α_j 's. For the sake of simplicity, we will assume that $L_{\mathbf{x}}$ has dimension n+1, but the attack still applies when the dimension is less than n+1. In other words, we assume that the \mathbf{x}_j 's and \mathbf{k} are linearly independent, which is a reasonable assumption since the \mathbf{x}_j 's are random. We now detail the three steps.

4.1 Disclosing the Hidden Lattice

The first step is based on the following observation, which is a simple consequence of (2):

Lemma 3. Let \mathbf{u} in \mathbb{Z}^m be orthogonal to \mathbf{b} . Then $\mathbf{p_u} = (\mathbf{u}.\mathbf{x}_1, \dots, \mathbf{u}.\mathbf{x}_n, \mathbf{u}.\mathbf{k})$ is orthogonal to the vector $\mathbf{v}_{\alpha} = (\alpha_1, \dots, \alpha_n, M)$.

Note that \mathbf{v}_{α} is independent of m, and so is the n-dimensional lattice $\mathbf{v}_{\alpha}^{\perp}$. We will see that, as m grows, most of the vectors of any reduced basis of the (m-1)-dimensional lattice \mathbf{b}^{\perp} are shorter and shorter. For such vectors u, the corresponding vectors \mathbf{p}_{u} are also shorter and shorter. But if \mathbf{p}_{u} gets smaller than $\Lambda(\mathbf{v}_{\alpha}^{\perp})$ (which is independent of m), then it is actually zero, that is, \mathbf{u} is orthogonal to all the \mathbf{x}_{i} 's and \mathbf{k} . This leads to the following condition:

Condition 4. Let $(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{m-1})$ be a reduced basis of \mathbf{b}^{\perp} . Then the first m - (n+1) vectors $\mathbf{u}_1, \dots, \mathbf{u}_{m-(n+1)}$ are orthogonal to each \mathbf{x}_j and \mathbf{k} .

One cannot expect that more than m-(n+1) vectors are orthogonal, because \bar{L}_x has dimension (n+1). If the condition is satisfied, the (n+1)-dimensional lattice $(\mathbf{u}_1,\ldots,\mathbf{u}_{m-(n+1)})^{\perp}$ contains each of the \mathbf{x}_j 's and \mathbf{k} . And one can see that it is in fact the lattice $\bar{L}_{\mathbf{x}}$, because they are orthogonal lattices of equal dimension, with one containing the other. Hence, the first step is as follows:

- 1. Compute a reduced basis $(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{m-1})$ of the orthogonal lattice \mathbf{b}^{\perp} .
- 2. Compute a basis of the orthogonal lattice $(\mathbf{u}_1, \dots, \mathbf{u}_{m-(n+1)})^{\perp}$ to obtain $\bar{L}_{\mathbf{x}}$.

This step is correct if and only if condition 4 is satisfied. We now precise in which case the condition is expected to hold. We first estimate the quantity $\Lambda(\mathbf{v}_{\alpha}^{\perp})$. If the α_j 's are uniformly distributed in [0..M-1], then $E(\alpha_j^2) \approx M^2/3$ so that $\|\mathbf{v}_{\alpha}\|$ is roughly $M\sqrt{n/3}$ (we assume the variance is negligible). With overwhelming probability, the gcd of all the α_j 's and M is equal to 1, implying that the lattice $\bar{\mathbf{v}}_{\alpha}$ is exactly \mathbf{v}_{α} , and therefore: $\det(\mathbf{v}_{\alpha}^{\perp}) = \|\mathbf{v}_{\alpha}\| \approx M\sqrt{n/3}$. Since the α_i 's are random, the n-dimensional lattice $\mathbf{v}_{\alpha}^{\perp}$ may be considered as random, so that:

$$\Lambda(\mathbf{v}^{\perp}) \approx \gamma \sqrt{n} \det(\mathbf{v}_{\alpha}^{\perp})^{1/n} \approx \gamma M^{1/n} (n/3)^{1/(2n)} \sqrt{n}.$$

We then estimate $\|\mathbf{p}_u\|$ for some well-chosen vectors \mathbf{u} . If the coordinates of the \mathbf{x}_j 's are independently uniformly distributed in $\{0,1\}$ (the case of the actual sparse distribution is discussed in section 4.4), and so are the α_j 's in [0..M-1], the expectation of the square of each coordinate of $\alpha_1\mathbf{x}_1 + \cdots + \alpha_n\mathbf{x}_n$ is roughly:

$$n \times \frac{1}{2} \times \frac{M^2}{3} + (n^2 - n) \times \frac{1}{4} \times \frac{M^2}{4} \approx \frac{1}{16}n^2M^2.$$

Hence $E(\|\mathbf{k}\|^2) \approx mn^2/16$, and we note that $E(\|\mathbf{x}_j\|^2) \approx m/2$. It follows that for any **u** (we again assume that the variance is negligible):

$$\|\mathbf{p}_{\mathbf{u}}\| \approx \|\mathbf{u}\| \sqrt{n \times m/2 + mn^2/16} \approx n\sqrt{m}\|\mathbf{u}\|/4.$$

Besides, we observe that the lattice \mathbf{b}^{\perp} contains a high-dimensional lattice of small determinant. Namely, it contains by (2) the (m-n-1)-dimensional lattice $(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{k})^{\perp}$, which determinant is less than $\|\mathbf{k}\| \times \prod_{j=1}^n \|\mathbf{x}_j\| \approx n\sqrt{m}(m/2)^{n/2}/4$. Hence, the vectors of any reduced basis of $(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{k})^{\perp}$ are expected to have norm around

$$\gamma \left[(m/2)^{n/2} n \sqrt{m} / 4 \right]^{1/(m-n-1)} \sqrt{m-n-1}.$$

Note that the expression is much smaller than $\gamma \|\mathbf{b}\|^{1/(m-1)} \sqrt{m-1}$ for large M, as $\|\mathbf{b}\| \approx M \sqrt{n}$. Therefore, the first m-n-1 vectors of any reduced basis of \mathbf{b}^{\perp} are likely to be short lattice points of $(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{k})^{\perp}$, and their expected length is given by the previous expression. For these vectors, the approximate length of the corresponding $\mathbf{p}_{\mathbf{u}}$ is:

$$\left[n\sqrt{m}/4 \times (m/2)^{n/2} \right]^{1/(m-n-1)} n\sqrt{m(m-n-1)}/4.$$

And condition 4 is likely to be satisfied if and only if this length is significantly smaller than $\Lambda(\mathbf{v}^{\perp})$, that is:

$$\left[n\sqrt{m}/4 \times (m/2)^{n/2} \right]^{1/(m-n-1)} n\sqrt{m(m-n-1)}/4 \ll M^{1/n}(n/3)^{1/(2n)} \sqrt{n}.$$

For sufficiently large m and n, the condition simplifies to:

$$\sqrt{mn(m-n-1)}/4 \ll M^{1/n} \tag{3}$$

The left-hand part is not large. In other words, this step is expected to succeed if the density $n/\log_2(M)$ is very small, so that $M^{1/n}$ is large.

4.2 Disclosing the Hidden Coefficients

In the second step, the lattice $\bar{L}_{\mathbf{x}}$ is known. The vectors \mathbf{x}_{j} 's are random and have entries in $\{0,1\}$, therefore these are short lattice points in $\bar{L}_{\mathbf{x}}$. Consider a short vector of some reduced basis of $\bar{L}_{\mathbf{x}}$. If its entries are all in $\{0,1\}$ or $\{0,-1\}$, it is very likely to be one of the $\pm \mathbf{x}_{j}$'s. Otherwise, its entries are probably in $\{0,\pm 1\}$, as it must be shorter than the \mathbf{x}_{j} 's. To get rid of these vectors, we transform the lattice $\bar{L}_{\mathbf{x}}$: we double all the lattice points, and we add the vector $(1,1,\ldots,1)$. The new lattice is:

$$L'_{\mathbf{x}} = 2\bar{L}_{\mathbf{x}} + \mathbb{Z} \times (1, 1, \dots, 1).$$

The vectors $2\mathbf{x}_i - (1, 1, ..., 1)$ belong to $L'_{\mathbf{x}}$, and their entries are ± 1 : they are short lattice points. We expect that there are no shorter vectors, since there is no obvious short combination of (1, 1, ..., 1) with the previous parasite vectors when doubled. In other words, the vectors $\pm [2\mathbf{x}_i - (1, 1, ..., 1)]$ should appear in any reduced basis of the lattice $L'_{\mathbf{x}}$. We expect this step to succeed if our lattice reduction algorithm provides a sufficiently reduced basis.

4.3 Recovering the Hidden Weights

Now that **k** and the \mathbf{x}_j 's are known, equation (2) reads as a modular linear system:

$$\mathbf{b} = \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \dots + \alpha_n \mathbf{x}_n \pmod{M}$$

The only unknowns are the α_j 's. If m is sufficiently large, this system is likely to have a unique solution. One way to solve this system is to use an orthogonal lattice. Denote by $x_{i,j}$ the j-th coordinate of \mathbf{x}_i . Also denote by b_i the i-th coordinate of \mathbf{b} . Let $m' \leq m$. Consider the lattice L generated by the rows of the following matrix:

$$\begin{pmatrix} b_1 & x_{1,1} & x_{2,1} & \dots & x_{n,1} & M & 0 & \dots & 0 \\ b_2 & x_{1,2} & x_{2,2} & \dots & x_{n,2} & 0 & M & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots & \ddots & \ddots & 0 \\ b_{m'} & x_{1,m'} & x_{2,m'} & \dots & x_{n,m'} & 0 & \dots & 0 & M \end{pmatrix}$$

We note that L^{\perp} must contain a point of the form $(-1, \alpha_1, \ldots, \alpha_n, ?, \ldots, ?)$, since the α_j 's satisfy the system. It follows that in any basis of L^{\perp} , there exists a linear combination (of the basis elements) for which the first coordinate is equal to -1. Such a combination can easily be computed from an extended gcd algorithm applied to the list formed by the first coordinate of each basis element. The element obtained is of the form $(-1, \beta_1, \ldots, \beta_n, ?, \ldots, ?)$. Clearly, the vector $(\beta_1, \ldots, \beta_n)$ modulo M satisfies the first m' solutions of the system. If m' is sufficiently large, it must be the unique solution $(\alpha_1, \ldots, \alpha_n)$. Hence, in order to solve the system, it suffices to compute a basis of the orthogonal lattice L^{\perp} , which can be done in polynomial time.

4.4 Sparse Hidden Subset Sums

If the hidden subset sum is sparse, that is $\kappa \ll n/2$, the condition (3) gets slightly better. Indeed, when one picks at most κ weights in each subset sum, one can show that $E(\|\mathbf{k}\|^2) \approx m\kappa^2/16$ and $E(\|\mathbf{x}_j\|^2) \approx m\kappa/n$. It follows, after a few computations, that the attack is expected to succeed if:

$$\left[\kappa\sqrt{m}/4 \times \sqrt{m}(m\kappa/n)^{n/2}\right]^{1/(m-n-1)} \kappa\sqrt{m(m-n-1)}/4 \ll M^{1/n}(n/3)^{1/(2n)}\sqrt{n}.$$

For sufficiently large m and n, the condition simplifies to:

$$\kappa \sqrt{m(m-n-1)}/4 \ll M^{1/n} \sqrt{n} \tag{4}$$

4.5 Affine Hidden Subset Sums

In the case of affine hidden subset sums, equation (2) becomes:

$$\mathbf{b} + s\mathbf{c} = \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \dots + \alpha_n \mathbf{x}_n + M\mathbf{k}$$
 (5)

Only \mathbf{b} , \mathbf{c} and \mathbf{M} are known. The attack can be adapted as follows. Clearly, lemma 3 remains correct if we take for \mathbf{u} a vector orthogonal to \mathbf{b} and \mathbf{c} . Step 1 thus becomes:

- 1. Compute a reduced basis $(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{m-2})$ of the orthogonal lattice $(\mathbf{b}, \mathbf{c})^{\perp}$.
- 2. Compute a basis of the orthogonal lattice $(\mathbf{u}_1, \dots, \mathbf{u}_{m-(n+1)})^{\perp}$ to obtain $\bar{L}_{\mathbf{x}}$.

The difference with the hidden subset problem is that, this time, the vector \mathbf{k} can be much bigger, due to the presence of s. More precisely, we have $s \approx M/2$ and $\|\mathbf{c}\| \approx M\sqrt{m/3}$, so that $\|\mathbf{k}\| \approx M\sqrt{m/12}$. In the appendix, we discuss how to modify the previous arguments to explain why the condition is still expected to be satisfied. Loosely speaking, when \mathbf{u} is short, the vector $\mathbf{p_u}$ cannot be guaranteed to be short, but all its entries except the last one are short, which suggests it cannot be a non-zero vector of $\mathbf{v}_{\alpha}^{\perp}$. Step 2 remains unchanged. And in step 3, we solve a similar linear system which is induced by (5). Therefore, the only difference when attacking affine hidden subset sums is that the underlying condition is less likely to be satisfied.

4.6 Experiments

We implemented our attack using the NTL [21] library version 3.1 developed by V. Shoup. We used two reduction algorithms: the LLL [11] algorithm to compute orthogonal lattices, and Schnorr's [17] Korkine-Zolotarev reduction algorithm with blocksize 20 to obtain better reduced bases. The implementation is fast: when $m \leq 300$ and M is no larger than 512 bits, the attack performed in less than 15 minutes on a 333MHz Ultrasparc-IIi. Heuristically, our attack works when the density is much smaller than 1, but only experiments can tell us what is exactly the limit. We stress that our implementation has not been optimized, which means that it might be possible to go a little bit further than what we obtained. For instance, one might try improved reduction algorithms such as [20]. In all our experiments, the attack worked as soon as step 1 was correct.

We first experimented the attack on hidden subset sums. If M is a 160-bit number (resp. 512-bit), the attack works up to $n \approx 45$ (resp. 90) with $m \approx 90$ (resp. 200). We were not able to attack larger values of n, even with larger values of m (up to 400). For affine hidden subset sums, results are not as good: if M is a 160-bit number (resp. 512-bit), the attack works up to $n \approx 35$ (resp. 60) with $m \approx 90$ (resp. 150). These results show that the conditions of validity for the attack which we gave previously are quite pessimistic. In particular, it appears that the attack is effective against small values of n, which are required in a smartcard environment. Analyzing table 1, we find that in the smartcard case, the HSS generator cannot be more efficient than the method of LL [12] for 160-bit and 512-bit exponents.

However, there is quite a gap between the largest parameters that our attack can handle and the parameters suggested in the scheme. When $M^{1/n}$ is very small, even very short vectors can be orthogonal to \mathbf{v}_{α} , so that step 1 is highly unlikely to succeed. This is for instance the case with $n = \log_2 M$. For such a

n, our attack cannot even exploit the sparsity of the subset sums, and the best attack remains the birthday attack. It follows that if one is willing to pay with storage by choosing a sufficiently large value of n to foil the attack, then one can choose a small κ to reduce significantly the computation time. This does not seem to be very useful in the 160-bit case, as LL's method offers very good performances. But it improves the situation for 512-bit exponents. Hence, the hidden subset sum generator appears to be useful only for server applications, with exponents of at least 512 bits.

5 The Randomness of the Hidden Subset Sum Generator

We analyze the distribution of the output of the hidden subset sum generator, and discuss its implications on the security of the scheme. For fixed M, the distribution is exponentially close (with respect to n) to the uniform distribution. We provide a proof in two cases: when the 0,1-coefficients are balanced, and when they are not. It was pointed out to us that such results were already known (technical result in [1], and a particular case is treated in [10]), but since our proof is quite different, we include it in appendix B. Our technique is based on the discrete Fourier transform, which might be of independent interest. The following result is proved in the extended version of [1] (Lemma 1, p12):

Theorem 5. There exists a c > 0 such that for all M > 0, if $\alpha_1, \ldots, \alpha_n$ are independently and uniformly chosen from [0..M-1], then the following holds with probability at least $1-2^{-cn}$:

$$\sum_{a=0}^{M-1} \left| P\left(\sum_{j=1}^{n} x_j \alpha_j = a\right) - \frac{1}{M} \right| \le 2^{-cn} \sqrt{M},$$

where P refers to a uniform and independent choice of the x_i 's in $\{0,1\}$.

This means that for fixed M, with overwhelming probability on the choice of the α_i 's, the distribution of the hidden subset sum generator is statistically close to the uniform distribution. And the result remains valid when one considers a polynomial number of samples instead of just one sample. More precisely, it is well-known that if for some distributions D_1 and D_2 , the statistical difference of D_1 and D_2 is less than ε , then the statistical difference of D_1^m and D_2^m is less than $m\varepsilon$, where the notation D^m means picking m elements independently at random from D. For instance, it can be proved by introducing hybrid distributions consisting of k elements picked from D_1 and m-k picked from D_2 .

Thus, for a fixed M, with overwhelming probability on the choice of the α_i 's, the distribution obtained by picking independently at random a polynomial number (in n) of outputs of the hidden subset sum generator corresponding to the α_i 's is statistically close to the uniform distribution. In particular, a polynomial-time adversary cannot distinguish the two distributions. But a successful attack against a scheme (for instance, DSS) using the hidden subset sum generator

would serve as a distinguisher for those distributions, assuming that the underlying scheme is unbreakable. Note that it was the case of our lattice-based attack.

Hence, the hidden subset sum generator is provably secure in this sense when the density is high. But this not very interesting from a practical point of view. Because when the density is high and the 0,1-coefficients are balanced, the scheme does not save over the obvious square-and-multiply generator. However, for the moment, we do not know what happens precisely with the actual distribution of the generator, that is, when the subset sums are sparse. Our technique is able to deal with the case of unbalanced coefficients (see section B.2), but we are unable to extend it to the sparse distribution. Maybe the technique of [1] will be more useful.

6 Conclusion

Boyko et al. proposed several methods to produce simple and very fast methods for generating randomly distributed pairs of the form $(x, q^x \mod p)$ and $(x, x^e \mod N)$ using precomputation. For discrete-log-based schemes, the security of these generators against active attacks relied on the presumed hardness of the hidden subset sum problem. We showed that the security against passive attacks relied on a variant of this problem, which we called the affine hidden subset sum problem. We provided a security criterion for these problems, based on the density. On the one hand, we presented an effective lattice-based algorithm which can heuristically solve these problems when the density is very small. Experiments have confirmed the theoretical analysis, and show that the practical interest of the proposed schemes is marginal. When applied to protocols such as DSS, ElGamal, or Schnorr, the proposed methods cannot be significantly more efficient on smartcards than traditional exponentiation methods based on precomputation, without being vulnerable to attacks. On the other hand, we showed that when the density is high, the distribution of the output of the generator was exponentially close to the uniform distribution, which provides undistinguishability against polynomial-time adversaries. The two conditions complement each other, but there is still a gap. It would be interesting to reduce the gap, either by improving the attack, or the hardness results. In particular, it would be nice to obtain a hardness result of practical interest for the actual hidden subset sum generator which uses sparse subset sums.

Acknowledgments. We would like to thank the anonymous referees for their helpful comments.

References

 M. Ajtai. Generating hard instances of lattice problems. In Proc. 28th ACM STOC, pages 99–108, 1996. Extended version at http://www.eccc.uni-trier.de/eccc/.

- 2. V. Boyko, M. Peinado, and R. Venkatesan. Speeding up discrete log and factoring based schemes via precomputations. In *Proc. of Eurocrypt' 98*, volume 1403 of *LNCS*, pages 221–235. Springer-Verlag, 1998.
- 3. E. Brickell, D.M. Gordon, K.S. McCurley, and D. Wilson. Fast exponentiation with precomputation. In *Proc. of Eurocrypt'92*, volume 658 of *Lecture Notes in Computer Science*, pages 200–207. Springer-Verlag, 1993.
- 4. E. F. Brickell and K. S. McCurley. An interactive identification scheme based on discrete logarithms and factoring. *Journal of Cryptology*, 5(1):29–39, 1992.
- M.J. Coster, A. Joux, B.A. LaMacchia, A.M. Odlyzko, C.-P. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992.
- 6. P. de Rooij. On the security of the Schnorr scheme using preprocessing. In *Proc.* of Eurocrypt'91, volume 547 of LNCS, pages 71–80. Springer-Verlag, 1991.
- 7. P. de Rooij. Efficient exponentiation using precomputation and vector addition chains. In *Proc. of Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 389–399. Springer-Verlag, 1995.
- 8. P. de Rooij. On Schnorr's preprocessing for digital signature schemes. *Journal of Cryptology*, 10(1):1–16, 1997.
- 9. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.
- R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.
- 11. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.
- 12. C.H. Lim and P.J. Lee. More flexible exponentiation with precomputation. In *Proc. of Crypto'94*, volume 839 of *Lecture Notes in Computer Science*, pages 95–107. Springer-Verlag, 1994.
- 13. National Institute of Standards and Technology (NIST). FIPS Publication 186: Digital Signature Standard, May 1994.
- P. Nguyen and J. Stern. Merkle-Hellman revisited: a cryptanalysis of the Qu-Vanstone cryptosystem based on group factorizations. In *Proc. of Crypto'97*, volume 1294 of *LNCS*, pages 198–212. Springer-Verlag, 1997.
- 15. P. Nguyen and J. Stern. Cryptanalysis of a fast public key cryptosystem presented at SAC '97. In *Proc. of SAC '98*, LNCS. Springer-Verlag, 1998.
- P. Nguyen and J. Stern. The Béguin-Quisquater server-aided RSA protocol from Crypto '95 is not secure. In *Proc. of Asiacrypt '98*, volume 1514 of *LNCS*, pages 372–379. Springer-Verlag, 1998.
- 17. C.-P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.
- 18. C.P. Schnorr. Efficient identification and signatures for smart cards. In *Proc. of Crypto'89*, volume 435, pages 239–252. Springer-Verlag, 1990.
- C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.
- 20. C.P. Schnorr and H.H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proc. of Eurocrypt'95*, volume 921 of *LNCS*, pages 1–12. Springer-Verlag, 1995.
- 21. V. Shoup. Number Theory C++ Library (NTL) version 3.6. Can be obtained at http://www.shoup.net/ntl/.

A The Condition for Affine Hidden Subset Sums

We explain why the modified step 1 of the attack is still expected to succeed against affine hidden subset sums when the density is very small. This time, the

vector \mathbf{k} is long:

$$\|\mathbf{k}\| \approx M\sqrt{m/12}.$$

Therefore, we no longer know a high-dimensional sublattice of $(\mathbf{b}, \mathbf{c})^{\perp}$ with small determinant. Still, we can hope that the first vectors of a reduced basis of $(\mathbf{b}, \mathbf{c})^{\perp}$ will have norm around $(\|\mathbf{b}\| \times \|\mathbf{c}\|)^{1/(m-2)} \sqrt{m-2} \approx (mM^2/3)^{1/(m-2)} \sqrt{m-2}$ which is small for large m. But the explanations of section 4.1 regarding condition 4 are no longer convincing, because $\mathbf{p}_{\mathbf{u}}$ cannot be guaranteed to be short, even if \mathbf{u} is very short (which it is). Recall that

$$\mathbf{p_u} = (\mathbf{u}.\mathbf{x}_1, \dots, \mathbf{u}.\mathbf{x}_n, \mathbf{u}.\mathbf{k}).$$

All the dot products $\mathbf{u}.\mathbf{x}_j$'s are still small, but the last coordinate $\mathbf{u}.\mathbf{k}$ might be large, since \mathbf{k} is long. However, we still expect that $\mathbf{p_u}$ cannot be a non-zero vector of $\mathbf{v}_{\alpha}^{\perp}$ if \mathbf{u} is very short, because most of its coordinates are very small.

To see this, let A be a bound for all the $|\mathbf{u}.\mathbf{x}_i|$'s, and B a bound for the last coordinate $[\mathbf{u}.\mathbf{k}]$. Denote by S the set of all vectors $(y_1,\ldots,y_{n+1})\in\mathbb{Z}^{n+1}$ where all the y_i 's are positive with $y_{n+1} \leq B$ and the remaining y_i 's are less than A. There are A^nB vectors in S. Now, consider the dot product of an element of S with \mathbf{v}_{α} . This dot product is in absolute value less than (nA+B)M, so that there are most 2(nA+B)M different possible values. It follows by the pigeon-hole principle that if $A^nB > 2(nA+B)M$, there must be a collision, that is, there must exist two distinct vectors \mathbf{z}_1 and \mathbf{z}_2 in S that have the same dot product with \mathbf{v}_{α} , which yields by difference an non-zero orthogonal vector to \mathbf{v}_{α} . The first n entries of this vector are less than A in absolute value, and the last entry is less than B in absolute value. This vector might be $\mathbf{p_u}$. But if $A^n B \ll 2(nA+B)M$, one does not expect any collision, and therefore $\mathbf{p_u}$ is unlikely to be a non-zero vector orthogonal to \mathbf{v}_{α} . The parameter B has limited influence on this condition, it is the value of A which is preponderant. In other words, when \mathbf{u} is short, $\mathbf{p}_{\mathbf{u}}$ is not necessarily short, but all its entries except the last one (which corresponds to \mathbf{k}) are small. This makes a small bound for A and a large one for B, and therefore, the condition $A^nB \ll 2(nA+B)M$ is nevertheless satisfied when the density is small.

B A Fourier Analysis of the Hidden Subset Generator

We compare the distribution of the output of the hidden subset sum generator with the uniform distribution, in two cases: when the 0,1-coefficients are balanced, and when they are not.

B.1 The Basic Case

Let M be an integer, and let $\alpha_1, \ldots, \alpha_n$ be independently and uniformly chosen from [0..M-1]. We first investigate the basic case where a linear combination

$$\sum_{j=1}^{n} x_j \alpha_j \pmod{M}$$

is produced with the x_j 's independently and uniformly chosen from $\{0,1\}$. We use the Fourier transform over the abelian group \mathbb{Z}_M . The characters $\chi_k(t) = e^{\frac{2\pi i kt}{M}}$ form an orthonormal basis of $L^2(\mathbb{Z}_M)$, endowed with the uniform probability measure and therefore any element f of $L^2(\mathbb{Z}_M)$ can be recovered from its Fourier coefficients $\hat{f}(k) = \frac{1}{M} \sum_{q=0}^{M-1} f(q) e^{-2\pi i kq/M}$, through the inverse Fourier formula:

$$f = \sum_{k=0}^{M} \hat{f}(k) \chi_k.$$

We now evaluate the expectation of each χ_k with respect to the image probability of the product probability over $\{0,1\}^m$ induced by the transformation: $(x_1,\ldots,x_n)\longmapsto \sum_{j=1}^n x_j\alpha_j$. We get for $k\neq 0$:

$$E(\chi_k) = E\left(e^{2\pi i k \sum_{j=1}^n x_j \alpha_j / M}\right) = \prod_{j=1}^n \frac{1}{2} \left(1 + e^{2\pi i k \alpha_j / M}\right).$$

Since $|1 + e^{i\theta}|^2 = (1 + \cos \theta)^2 + \sin^2 \theta = 2 + 2\cos \theta$, it follows that:

$$\sum_{k=1}^{M-1} |E(\chi_k)|^2 = \sum_{k=1}^{M-1} \prod_{j=1}^n \frac{1 + \cos(2\pi k\alpha_j/M)}{2}.$$

We estimate this expression, with respect to a uniform choice of the α_j 's in [0..M-1]:

Lemma 6. Let k be an integer in [1..M-1]. If α is a random integer uniformly chosen from [0..M-1] then:

$$E\left[\cos(2k\pi\alpha/M)\right] = E\left[\sin(2k\pi\alpha/M)\right] = 0.$$

Proof. Let $\theta = 2k\pi/M$. By definition, the two expectations are respectively the real and imaginary part of: $E = \frac{1}{M} \sum_{\alpha=0}^{M-1} e^{i\alpha\theta}$. But since $k \in [1..M-1]$, the complex $e^{i\theta}$ is an M-th root of unity different from 1. Therefore the geometric sum is actually equal to zero, which completes the proof.

Corollary 7. Let $\varepsilon > 0$. If the α_j 's are independently and uniformly chosen from [0..M-1], then the following holds with probability at least $1-\varepsilon$:

$$\sum_{k=1}^{M-1} \prod_{j=1}^{n} \frac{1 + \cos\left(2\pi k \alpha_j / M\right)}{2} \le \frac{M 2^{-n}}{\varepsilon}.$$

Proof. Denote by X the left-hand random variable. By independence of the α_j 's, the previous lemma shows that:

$$E(X) = \sum_{k=1}^{M-1} \prod_{i=1}^{n} \frac{1}{2} \le \frac{M}{2^n}.$$

And the result follows by Markov inequality.

Now assume that the α_j 's satisfy the inequality of the previous proposition for some $\varepsilon > 0$. Then:

$$\sum_{k=1}^{M-1} |E(\chi_k)|^2 \le \frac{M2^{-n}}{\varepsilon}.$$

This in turn means that $f - \hat{f}(0) = \sum_{k=1}^{M} \hat{f}(k) \cdot \chi_k$ has expectation bounded by $||f|| \cdot \sqrt{M2^{-n}/\varepsilon}$, where ||f|| denotes the L^2 -norm of f with respect to the uniform probability on \mathbb{Z}_M . This reads:

$$|E(f) - \hat{f}(0)| \le ||f|| \cdot \sqrt{M2^{-n}/\varepsilon},$$

and establishes a bound on the difference between the expectation E(f) and the expectation of the same function f taken over the uniform probability on \mathbb{Z}_M .

Applying to a given event X, and using corollary 7, we obtain:

Theorem 8. Let $\varepsilon > 0$. If $\alpha_1, \ldots, \alpha_n$ are independently and uniformly chosen from [0..M-1], then the following holds with probability at least $1-\varepsilon$: for any subset X of \mathbb{Z}_M with uniform probability δ , the probability δ' of X with respect to the image probability of the product probability over $\{0,1\}^m$ induced by the transformation $(x_1,\ldots,x_n)\longmapsto \sum_{j=1}^n x_j\alpha_j$, differs from δ by an amount bounded by:

 $\sqrt{\delta M 2^{-n}/\varepsilon}$.

B.2 The Case of Unbalanced Coefficients

We now assume that the coefficients x_j 's are unbalanced and chosen according to a distribution where zeros are more likely to appear. We consider the probability ditsribution on $\{0,1\}$ where one receives probability p and zero 1-p and we endow $\{0,1\}^n$ with the product probability P_p . It is easy to show that the results of the previous section go through mutatis mutandis. The main difference is that the expectation of χ_k becomes

$$\prod_{i=1}^{n} (1-p) + pe^{\frac{2\pi i k \alpha_j}{M}}$$

An easy computation shows that this amounts to replacing the term $\frac{1}{2}[1 + \cos(2\pi k\alpha_j/M)]$ by $(1-p)^2 + 2p(1-p)\cos(2\pi k\alpha_j/M) + p^2$ in the expression of $|E(\chi_k)|^2$. Lemma 6 shows that the cosine term has zero mean, with respect to a uniform choice of α_j from [0..M-1]. It follows that the previous term has expectation equal to $(1-p)^2 + p^2$. We finally get:

Theorem 9. Let $\varepsilon > 0$. If $\alpha_1, \ldots, \alpha_n$ are independently and uniformly chosen from [0..M-1], then the following holds with probability at least $1-\varepsilon$: for any subset X of \mathbb{Z}_M with uniform probability δ , the probability δ' of X, with respect to the image probability of P_p induced by the transformation $(x_1, \ldots, x_n) \mapsto \sum_{j=1}^n x_j \alpha_j$, differs from δ by an amount bounded by

$$\sqrt{\delta M((1-p)^2+p^2)^n/\varepsilon}$$
.

Information-Theoretic Cryptography

(Extended Abstract)

Ueli Maurer*

Department of Computer Science Swiss Federal Institute of Technology (ETH), Zurich CH-8092 Zurich, Switzerland maurer@inf.ethz.ch

Abstract. We discuss several applications of information theory in cryptography, both for unconditional and for computational security. Unconditionally-secure secrecy, authentication, and key agreement are reviewed. It is argued that unconditional security can practically be achieved by exploiting the fact that cryptography takes place in a physical world in which, for instance due to noise, nobody can have complete information about the state of a system.

The general concept of an information-theoretic cryptographic primitive is proposed which covers many previously considered primitives like oblivious transfer, noisy channels, and multi-party computation. Many results in information-theoretic cryptography can be phrased as reductions among such primitives We also propose the concept of a generalized random oracle which answers more general queries than the evaluation of a random function. They have applications in proofs of the computational security of certain cryptographic schemes.

This extended abstract summarizes in an informal and non-technical way some of the material presented in the author's lecture to be given at Crypto '99.

Key words: Information theory, unconditional security, conditional independence, information-theoretic primitive, generalized random oracle.

1 Introduction

Historically, information theory and cryptography are closely intertwined, although the latter is a much older discipline. Shannon's foundation of information theory [40] was motivated in part by his work on secrecy coding during the second world war, and it may be for this reason that his work was not de-classified until 1948 when his seminal paper was published. His 1949 companion paper on the communication theory of secrecy systems [39] was, like Diffie and Hellman's later discovery of public-key cryptography [19], a key paper in the transition of cryptography from an art to a science.

There are two types of cryptographic security. The security of a cryptographic system can rely either on the computational infeasibility of breaking it (computational security), or on the theoretical impossibility of breaking it, even

^{*} Supported in part by the Swiss National Science Foundation, grant no. 20-42105.94.

using infinite computing power (information-theoretic or unconditional security). Because no computational problem has been proved to be computationally difficult for a reasonable model of computation, the computational security of every practical cryptographic system relies on an unproven intractability assumption. In contrast, information-theoretically secure systems rely on no such assumptions, but they rely on an assumption about the probabilistic behavior of the universe, for instance of a noisy channel or a quantum measurement. However, even computationally-secure systems rely on such assumptions, at least the tacitly made assumption that random keys can be generated and that they are independent of an adversary's entire a priori knowledge.

While information-theoretic security is stronger than computational security, it is usually less practical. In fact, Shannon's proof that perfect secrecy requires a secret key of the same length as the plaintext is often taken as evidence that unconditional security can never be practical. However, this precipitate jump to conclusions should be reconsidered: in contrast to Shannon's model, in which his result holds, cryptography takes place in a physical world (every communication channel is based on a physical process) in which nobody can have complete information about the state of a system, for instance due to noise or theoretical limitations of quantum physics.

Information theory has several applications in cryptography. First, it allows to prove the unconditional security of cryptographic systems. Second, it allows to prove impossibility and lower bound results on the achievability of unconditional security. Third, it is a key tool in reduction proofs showing that breaking a cryptographic system is as hard as breaking an underlying cryptographic primitive (e.g. a one-way function or a pseudo-random function).

In this extended abstract we give an overview of known applications and results of information theory in cryptography. Due to space limitations we cannot give a complete overview of the extensive literature on the subject. The treatment is informal and non-technical, emphasizing concepts and general viewpoints. In Section 2, we review some basic concepts of information theory and state two basic facts on conditional independence. In Section 3, we summarize known results on unconditional secrecy, authentication, and key agreement. In Section 4 we take a general approach to cryptographic primitives and reductions among them. The concept of generalized random oracles is sketched briefly in Section 5, followed by some conclusions.

2 Random Variables, Entropy, and Conditional Independence

Information theory, like statistics, is a mathematical theory based on probability theory. In almost all applications of probability theory in cryptography one considers a discrete random experiment which is conceptually very simple: it is defined by a finite or countably infinite set called the sample space, consisting

¹ We refer to [7] and [13] for a more detailed introduction to information theory, and to [21] for an introduction to probability theory.

of all elementary events, and a *probability measure* assigning a non-negative real number to every elementary event, such that the sum of all these probabilities is equal to 1. An *event* of a discrete random experiment is a subset of the sample space, and the probability assigned to it is the sum of the probabilities of its elementary events.

A discrete random variable X is a mapping from the sample space to a certain range \mathcal{X} and is characterized by its probability distribution P_X that assigns to every $x \in \mathcal{X}$ the probability $P_X(x)$ of the event that X takes on the value x.

The *entropy* of a random variable X is a real number that measures the uncertainty about the value of X when the underlying random experiment is carried out. It is defined as

$$H(X) = -\sum_{x} P_X(x) \log_2 P_X(x),$$

assuming here and in the sequel that terms of the form $0 \log 0$ are excluded from the summation. The particular formula will be irrelevant below, but we need certain important properties of entropy. It is easy to verify that

$$0 \le H(X) \le \log_2 |\mathcal{X}|$$

with equality on the left if and only if $P_X(x) = 1$ for some $x \in \mathcal{X}$ and with equality on the right (for finite \mathcal{X}) if and only if $P_X(x) = 1/|\mathcal{X}|$ for all $x \in \mathcal{X}$.

The deviation of the entropy H(X) from its maximal value can be used as a measure of non-uniformity of the distribution P_X . While there are other such non-uniformity measures (e.g., based on Rényi entropy and min-entropy, which have some interesting applications not discussed in this paper), the significance of Shannon entropy is that it satisfies some intuitive rules (e.g., the chain rule) and that it gives the right answer to fundamental questions in communication engineering: how much can (reversible) data compression reduce the size of a message, and how many information bits per channel use can be transmitted reliably over a given noisy communication channel?

When several random variables (e.g. X,Y,Z with joint distribution P_{XYZ}) are considered, they are always defined on the same random experiment. The definition of H(X) can be generalized to the definition of the joint entropy of two or more random variables. For instance, we have $H(XYZ) = -\sum_{(x,y,z)} P_{XYZ}(x,y,z) \log P_{XYZ}(x,y,z)$.

The conditional probability distribution $P_{X|Y}(\cdot,y)$ of the random variable X, given the event Y=y, is defined by $P_{X|Y}(x,y)=P_{XY}(x,y)/P_{Y}(y)$ when $P_{Y}(y)\neq 0$. For every such $y\in \mathcal{Y}$, $P_{X|Y}(\cdot,y)$ is a probability distribution satisfying $\sum_{x\in\mathcal{X}}P_{X|Y}(x,y)=1$. The entropy of this distribution is the conditional entropy of X, given the event Y=y:

$$H(X|Y = y) = -\sum_{x} P_{X|Y}(x, y) \log_2 P_{X|Y}(x|y).$$

The conditional uncertainty of X, given the random variable Y, is defined as the average over all y of H(X|Y=y), and is not the entropy of a distribution:

$$H(X|Y) = \sum_{y \in \mathcal{Y}: P_Y(y) \neq 0} H(X|Y = y) P_Y(y).$$

One can show that additional knowledge can never increase entropy:

$$0 \leq H(X|Y) \leq H(X),$$

with equality on the left if and only if Y determines X (except when $P_Y(y) \neq 0$) and with equality on the right if and only if X and Y are statistically independent (see below).

An important rule for transforming entropies is

$$H(XY) = H(X) + H(Y|X),$$

i.e., the joint entropy about X and Y is the entropy about X plus the additional entropy about Y, given that X is known. This so-called chain rule can be used repeatedly to expand $H(X_1X_2\cdots X_N)$ as

$$H(X_1 X_2 \cdots X_N) = \sum_{n=1}^{N} H(X_n | X_1 \cdots X_{n-1}).$$

Note that the order in which variables are extracted is arbitrary. For example,

$$H(XYZ) = H(X) + H(Y|X) + H(Z|XY)$$

= $H(Y) + H(Z|Y) + H(X|YZ)$.

The mutual information I(X;Y) between two random variables X and Y is defined as the amount by which the uncertainty (entropy) about X is reduced by learning Y:

$$I(X;Y) = H(X) - H(X|Y).$$

The term mutual stems from the fact that, as can easily be verified, I(X;Y) = I(Y;X) = H(Y) - H(X|Y). The conditional mutual information between X and Y, given the random variable Z, is defined as

$$I(X;Y|Z) = H(X|Z) - H(X|YZ).$$

We have I(X;Y|Z) = 0 if and only if X and Y are statistically independent when given Z.

Conditional independence is a fundamental concept in information-theoretic cryptography. Two events A and B are statistically independent, here denoted [A;B], if $P(A \cap B) = P(A) \cdot P(B)$. In the following we will drop the symbol \cap and use the shorter notation P(A,B) or simply P(AB) for $P(A \cap B)$. Two events A and B are conditionally independent, given the event C, denoted [A;B|C], if $P(A \cap B \cap C) \cdot P(C) = P(A \cap C) \cdot P(B \cap C)$ or, in our short notation,

$$P(ABC) \cdot P(C) = P(AC) \cdot P(BC).$$

If P(C) > 0, this is equivalent to $P(AB|C) = P(A|C) \cdot P(B|C)$. Note that independence is symmetric, i.e. $[A;B] \iff [B;A]$ and $[A;B|C] \iff [B;A|C]$. Let \overline{A} denote the complement of the event A. One can also show that $[A;B] \iff [A;\overline{B}]$ and $[A;B] \iff [A;\overline{B}]$ while $[A;BC] \implies [A;\overline{B}C]$ is false in general.

The concept of statistical independence and this notation can be extended to a situation where any of A, B and C can be either an event or a random variable. Independence when random variables are involved means that the independence relation holds when any random variable is replaced by the event that it takes on a particular value. For instance, if A is an event and X and Y are random variables, then [X;A|Y] is equivalent to [X=x;A|Y=y] for all x and y.

The following theorem stated without proof implies the rules for a calculus of conditional independence and is useful for simplifying certain security proofs. It states under which condition an event or random variable can be added to an independence set. Moreover, any random variables in an independence set can be dropped, and, if accompanied in the set only by other random variables, then it can also be moved to the conditioning set.

Theorem 1. Let S, T, U and V each be an event or a random variable (defined for the same random experiment). Then

$$[S;T|V]$$
 and $[S;U|TV] \implies [S;TU|V]$.

If U is a random variable, then $[S; TU|V] \Longrightarrow [S; T|V]$, and if also T is a random variable, then $[S; TU|V] \Longrightarrow [S; U|TV]$

Note that if S, T, and U are events, then $[S; TU] \Longrightarrow [S; T|U]$ and $[S; TU] \Longrightarrow [S; T]$ are false in general. For instance, let P(S) = P(T) = P(U) = 0.5, P(ST) = P(SU) = P(TU) = 0.2, and P(STU) = 0.1. Then P(STU) = P(S)P(TU) = 0.1 but $P(STU)P(U) = 0.05 \neq P(SU)P(TU) = 0.04$.

3 Unconditional Secrecy, Authenticity, and Key Agreement

One of the fundamental problems in cryptography is the transmission of a message M from a sender Alice to a receiver Bob such that an adversary Eve with access to the communication channel is unable to obtain information about M (secrecy). Moreover, if Eve has write-access to the channel, then Bob must not accept a fraudulent message modified or inserted by Eve (authenticity). This is achieved by Alice and Bob sharing a secret key K used together with the message to compute a ciphertext C to be transmitted over the channel. The security can be either computational or information-theoretic, and we are here only interested in the latter.

For instance, C is an encryption of M, or M together with an appended message authentication code.

3.1 Unconditional Authentication

Unconditionally secure message authentication based on a shared secret key was first considered in [24] and later in a large number of papers (e.g., see [44], [41], [42]). Another line of research is devoted to proving lower bounds on the cheating probability as a function of the entropy of the key, H(K); see [33] for a discussion and generalization of these bounds. Assume that the secret key K is used to authenticate a message M, resulting in ciphertext C, and let p_I and p_S denote Eve's probability of successfully creating a fraudulent message (impersonation attack) and of successfully replacing a valid message by a fraudulent message (substitution attack), respectively, then the following lower bounds hold for any authentication system, for an optimal cheating strategy:

$$p_I \ge 2^{-I(C;K)}, p_S \ge 2^{-H(K|C)}, \text{ and } \max(p_I, p_S) \ge 2^{-H(K)/2}.$$

In other words, half of the key must be used for protecting against an impersonation attack, and the other half to prevent a substitution attack. These bounds can be generalized in various directions, for instance to a setting where n consecutive messages are authenticated using the same key. Then the cheating probability is lower bounded by $2^{-H(K)/(n+1)}$.

This bound can easily be achieved, when the message space is a subset of the k-bit strings, by a scheme based on polynomial evaluation (where the secret key consists of the n+1 coefficients of a polynomial over $GF(2^k)$ of degree n), achieving cheating probability 2^{-k} . One can show that equality in the bounds cannot be achieved for larger message spaces. However, Gemmell and Naor [23] proved that interactive protocols for authenticating a k-bit message can make more efficient use of the secret key than non-interactive protocols.

3.2 Unconditional Secrecy

It is well-known that the one-time pad [43] provides perfect secrecy (though no authenticity unless the message is redundant), where perfect secrecy is the strongest possible type of security of an encryption system and is defined as the message M and the ciphertext C being statistically independent: I(M;C)=0, or [M;C]. Shannon [39] proved that for every perfect system, $H(K) \geq H(M)$, i.e. perfect secrecy requires an impractically large amount of secret key. A system that is perfect for every distribution P_M of the message M is called robustly perfect. The (binary) key of such a system must be at least as long as the message; hence the one-time pad is optimal. Rather than proving Shannon's bound, we look at a more general setting below from which Shannon's result follows as a special case.

In the following we assume that an insecure communication channel between Alice and Bob is available. Since we are interested in results on security and not primarily on communication efficiency, this assumption is made without loss of generality. It implies that a secure key agreement protocol implies a secure encryption scheme (e.g. using the one-time pad), and the reverse implication is trivial. Thus we can restrict our attention to key agreement.

3.3 Unconditional Key Agreement: Impossibility Results and Bounds

Definition 1. A key-agreement protocol consists of a communication phase in which Alice and Bob alternate sending each other messages C_1, C_2, C_3, \ldots , where we assume that Alice sends messages C_1, C_3, C_5, \ldots and Bob sends messages C_2, C_4, C_6, \ldots Each message can depend on the sender's entire view of the protocol and possibly on privately generated random bits.

After the communication phase, Alice and Bob each either accepts or rejects the protocol execution, depending on whether he or she believes to be able to generate a shared secret key. If Alice accepts, she generates a key S depending on her view of the protocol.

Similarly, if Bob accepts, he generates a key S' depending on his view of the protocol. Even if a party does not accept, he or she may generate a key. \diamond

In the sequel we assume without loss of generality that S and S' are binary strings of length |S| = |S'| = k, where the goal is of course to make k as large as possible. Let t be the total number of messages and let $C^t = [C_1, \ldots, C_t]$ denote the set of exchanged messages.

Informally, a key agreement protocol is secure if the following conditions are satisfied [34,45]:

- whenever Eve is only passive, then Alice and Bob both accept, and
- whenever one of the parties accepts, then
 - the other party has also generated a key (with or without accepting), and the two keys agree with very high probability (i.e. $P[S \neq S'] \approx 0$),
 - the key S is very close to uniformly distributed, i.e. H(S) (and hence also H(S')) is very close to k, and
 - Eve's information about S, $I(S; C^t Z)$, given her entire knowledge, is very small (see the definition of Z below).

It appears obvious that if Alice and Bob do not share at least some partially secret information initially, they cannot generate an information-theoretically secure secret key S (i.e. H(S) = 0) if they can only communicate over a public channel accessible to Eve, even if this channel is authenticated.³ This follows from inequality (1) below and implies Shannon's bound $H(K) \geq H(M)$.

In order for the key agreement problem to be interesting and relevant, we therefore need to consider a setting that takes into account the possibility that Alice and Bob each have some correlated side information about which Eve does not have complete information. Several such scenarios, practical and theoretical, have been considered. For instance, Fischer and Wright analyzed a setting in which Alice and Bob are assumed to have been dealt disjoint random deals of cards. More natural and realistic scenarios may arise from exploiting an adversary's partial uncertainty due to unavoidable noise in the communication channel or intrinsic limitations of quantum physics. We refer to [4] for a discussion of

³ This fact can be rephrased as follows: There exists no unconditionally-secure public-key cryptosystem or public-key distribution protocol.

quantum key agreement. The problem of designing information-theoretically secure key agreement protocols is hence to identify practical scenarios in which the adversary's total information can be bounded, and then to design a protocol that exploits this scenario.

Such a scenario can generally be modeled by assuming that Alice, Bob, and Eve initially know random variables X, Y, and Z, respectively, which are jointly distributed according to some probability distribution P_{XYZ} .⁴ These random variables could be the individual received noise signals of a deep-space radio source, or the individual received noisy versions of a random bit string broadcast by a satellite at a very low signal power.

For this setting it was shown in [35] that⁵

$$H(S) \leq I(X; Y \downarrow Z),$$

where $I(X; Y \downarrow Z)$ denotes the intrinsic conditional mutual information between X and Y, given Z, which is defined as follows [35]:

$$I(X;Y\!\downarrow\! Z) := \inf_{P_{\overline{Z}\mid Z}} \; \left\{ I(X;Y|\overline{Z}) \; : \; P_{XY\overline{Z}} = \sum_{z\in\mathcal{Z}} P_{XYZ} \cdot P_{\overline{Z}\mid Z} \right\} \; .$$

The above inequality is a generalization of the following bound [32]:

$$H(S) \le \min[I(X;Y), I(X;Y|Z)]. \tag{1}$$

Note that I(X;Y) and I(X;Y|Z) are obtained when \overline{Z} is a constant or $\overline{Z}=Z$, respectively, and that $I(X;Y|Z) \geq I(X;Y)$ is possible.

3.4 Unconditional Key Agreement by Public Discussion

The previous bound is an impossibility result. In order to prove constructive results about key agreement by public discussion, we need to make an explicit assumption about the distribution P_{XYZ} . A very natural assumption, which is often made in an information-theoretic context, is that the same random experiment generating X, Y, and Z is repeated independently many times. One can then define the secret key rate S(X;Y||Z) [32] as the maximum rate (per executed random experiment) at which Alice and Bob can generate secret key, assuming (for now) an authenticated but otherwise insecure communication channel.

This rate turns out to be positive even in cases where intuition might suggest that key agreement is impossible. For instance, when a satellite broadcasts random bits and X, Y, and Z are the bits (or more generally signals) received

⁴ More generally, the distribution P_{XYZ} could be under Eve's partial control and may only partly be known to Alice and Bob, for instance in the case of a quantum transmission disturbed by Eve.

⁵ neglecting here the fact that the bound can be slightly greater if imperfect secrecy or a non-zero failure probability is tolerated.

by Alice, Bob, and Eve, respectively, then key agreement is possible under the sole condition that Eve's channel is not completely perfect, even if Alice's and Bob's channels are by orders of magnitude more noisy than Eve's channel, for instance when Alice's and Bob's bit error rates are very close to 50% (e.g. 0.499) and Eve's bit error rate is very small (e.g. 0.001).

We conjecture that the secret key rate S(X;Y||Z) is positive if and only if $I(X;Y\downarrow Z)$ is positive, and the two quantities may even be equal. Even if the public discussion channel is *not* authenticated, key agreement is still possible. The secret key rate is even equal to S(X;Y||Z) (where an authenticated channel is assumed) [34,45], except if Eve can either generate from Z a random variable \tilde{Y} such that $P_{X\tilde{Y}} = P_{XY}$ or, symmetrically, a random variable \tilde{X} such that $P_{\tilde{X}Y} = P_{XY}$. In both these cases, key agreement is impossible.

Many results on unconditionally secure key agreement were recently refined in various ways. We refer to [46] for a very good overview and to [9,47] for detailed accounts of recent results in unconditionally-secure key agreement.

3.5 Public Randomness and Memory-Bounded Adversaries

In this section we briefly discuss two other scenarios in which Eve cannot obtain complete information, and where this can be exploited by Alice and Bob to agree on a very long unconditionally-secure secret key S (e.g. 1 Gigabyte), assuming that they share only a short secret key K (e.g. 5000 bits) initially.

Suppose that all parties, including Eve, have access to a public source of randomness (similar to a random oracle) which is too large to be read entirely by Eve in feasible time. Then Alice and Bob can access only a moderate number of random bits, selected and combined using the secret key, such that unless Eve examines a substantial fraction (e.g. one half) of the random bits (which is infeasible), she ends up having no information about the generated key [30]. More precisely, there exists an event \mathcal{E} such that $[S;KW|\mathcal{E}]$, where W summarizes Eve's entire observation resulting from an adaptive access strategy. This is true even if Eve is given access to the secret key K after finishing her access phase. Moreover, for any adaptive strategy (without knowledge of K), the event \mathcal{E} has probability exponentially close to 1. In other words, conditioned on this high-probability event, the scheme achieves perfect secrecy.

While the original proof assumed that Eve accesses individual bits of the source, Aumann and Rabin [37] showed that the scheme is secure even if she accesses arbitrary bits of information about the random bits, e.g. Boolean functions evaluated on all the randomizer bits.

This also motivates the following model [11]: Alice and Bob publicly exchange a random string too long to fit into Eve's memory, and use a scheme similar to that described above, based on a short initially shared secret key. It's security holds based on the sole assumption that Eve's memory capacity is bounded, without assumption about her computing power.

4 Information-Theoretic Primitives: A General Perspective

In both complexity-theoretic and information-theoretic cryptography, an important body of research is devoted to the reduction of one primitive to another primitive, e.g. of a pseudo-random number generator to a one-way function [27] or of oblivious transfer to the existence of a noisy channel [16]. In this section we informally define a general notion of an information-theoretic cryptographic primitive, discuss the general reduction problem among such primitives and possible goals of such a reduction, and show that many results in the literature fit into this general framework.

4.1 Definition of IT-Primitives

Definition 2. A (stateless) information-theoretic cryptographic primitive (IT-primitive or simply primitive, for short) is an abstractly defined mechanism (which can be viewed as a service offered by a trusted party) to which $n \geq 2$ players P_1, \ldots, P_n have access. For every invocation of the primitive, each player P_i can provide a (secret) input X_i from a certain domain and receives a (secret) output Y_i from a certain range according to a certain (usually publicly known) conditional probability distribution $P_{Y_1,\ldots,Y_n|X_1,\ldots,X_n}$ of the outputs, given the inputs.

As described, different invocations of the primitive are independent, but more generally an IT-primitive can have an internal state: the players can provide inputs and receive outputs in consecutive rounds, where in each round the dependence of the outputs on the inputs and the current state is specified by a conditional probability distribution.

This concept encompasses as special cases a very large class of previously considered cryptographic primitives, including secure message transmission, noisy channels, all types of oblivious transfer, broadcast channels, and secure multiparty computation, as will be explained below. The concept of a secure reduction of one primitive to another primitive will be discussed later.

There are at least two different ways of defining what it means for the players to have incomplete knowledge of the distribution $P_{Y_1,...,Y_n|X_1,...,X_n}$.

- The distribution can be any one in a class of distributions, possibly chosen by an adversary. If such a primitive is used in a protocol, security must be guaranteed for all distributions in the class.
- The distribution is fixed, but some or all players' knowledge about the distribution may be incomplete. This can be modeled by letting each player receive an additional output summarizing his information about the distribution. This extra output can be viewed as part of the regular output and hence this case is covered by the above definition.

4.2 Examples

Let us first consider some IT-primitives for n=2 players, called Alice and Bob.

- A noisy channel from Alice to Bob: Alice can provide an input X from a certain domain (e.g. a bit) and Bob receives the output Y generated according to the conditional distribution $P_{Y|X}$. For instance, in a binary symmetric channel with error rate ϵ , $P_{Y|X}(y,x) = \epsilon$ if $x \neq y$ and $P_{Y|X}(y,x) = 1 \epsilon$ if x = y. The (γ, δ) -unfair noisy channel of [18] is a binary symmetric channel with error probability chosen by the adversary in the interval $[\gamma, \delta]$.
- Oblivious transfer (OT) of any type is a classical example of an IT-primitive. In standard 1-out-of-2 OT, Alice chooses as input two bits (or bit strings), Bob chooses as input a selection bit, and Bob learns as output the corresponding bit (or string), while Alice learns nothing. In the generalized OT of Brassard and Crépeau [8], Bob can choose to receive any binary function of the two input bits, and this can be generalized further [8,10] to allow Bob to specify any channel over which he receives the two input bits, with the only constraint that his uncertainty (entropy) about the two input bits be at least γ , for some $0 < \gamma < 2$.
- Damgård, Kilian, and Salvail [18] introduced a more general two-party primitive which they called weak oblivious transfer (WOT). In this model, also Alice receives an output that gives her partial information about Bob's choice.
- A commitment scheme is a primitive with state: Alice inputs a value (which
 is kept as the state). Later, upon initiation of an opening phase, Alice chooses
 (with a binary input) whether or not she agrees to open the commitment,
 and Bob's output is the committed value or a dummy value, respectively.
 This primitive can also be defined with respect to several players (see below).

Damgård et al. [18] also introduced a two-player primitive called weak generic transfer (WGT) that is similar to the two-player case of our general IT-primitive. However, the models differ in the following way: In WGT, cheating by one of the players is modeled by an additional output which the player receives only when cheating, but not otherwise. Passive cheating means that the player collects this information, without deviating from the protocol, and active cheating means that the player can take this extra information into account during the execution of the protocol. In contrast, cheating is in our definition not considered as part of the primitive, but as misbehavior to be protected against in a protocol in which the primitive is used. The possible assumption that a player receives additional information when cheating can be phrased as a security condition of a protocol in which the primitive is used.

Next we consider primitives for n = 3 players which we call Alice, Bob, and Eve, and where it is known in advance that Alice and Bob are honest while Eve is a cheating adversary.

– Key agreement between Alice and Bob: The players have no input⁶ and receive outputs Y_A, Y_B , and Y_E , respectively. The specification of the primitive

⁶ Invocation of the primitive could actually be considered as a special type of input.

- is that Y_A is chosen uniformly at random from the key space, that $Y_B = Y_A$, and that Y_E is some dummy output that is independent of Y_A .
- A noisy random source: there are again no inputs, but the outputs Y_A, Y_B , and Y_E are selected according to a general distribution. This corresponds to the key agreement scenario discussed in the previous section, where the random variables Y_A, Y_B , and Y_E were called X, Y, and Z and generated according to some distribution P_{XYZ} . The distribution of a random deal of cards [22] is also a special case.
- Wyner's wire-tap channel [48] and the generalization due to Csiszár and Körner [17]: A symbol sent by Alice is received by Bob and Eve over two dependent noisy channels.
- Secure message transmission also fits into this framework: Alice's input is received by Bob, Eve receives no output. If the channel is authenticated but not confidential, then Eve also receives the output.
- A quantum channel from Alice to Bob can also be modeled as an IT-primitive
 if the eavesdropper is forced to measure each quantum state independently.
 For modeling the possibility that Eve could perform general quantum computations on the quantum states, our IT-primitive can be generalized to the
 quantum world.

We now describe some IT-primitives for general n:

- A broadcast channel: A designated sender provides an input which is received (consistently) by all other n-1 players.
- Secure function evaluation for an agreed function: each player provides a secret input and receives the output of a function evaluated on all inputs.
 The players' output functions can be different.
- Secure multi-party computation [26,3,12] among a set of $n \geq 2$ players: here the primitive keeps state. This corresponds to the general paradigm of simulating a trusted party.
- A random oracle can also be interpreted in this framework.

4.3 Reductions among IT-Primitives

The general reduction problem can be phrased as follows: assuming the availability of one primitive G (more generally, several primitives G_1, \ldots, G_s), can one construct primitive H, even if some of the players cheat, where the type of tolerable cheating must be specified. One can distinguish many types of cheating. Three well-defined cases are active cheater(s) who deviate from the protocol in an arbitrary manner, passive cheaters who follow the protocol but record all information in order to violate other players' privacy, and fail-corrupted cheaters who may stop executing the protocol at any time. Cheating players are usually modeled by assuming a central adversary who may corrupt some of the players.

One generally assumes without essential loss of generality or applicability that insecure communication channels between any pair of entities are available.

Such information-theoretic reductions among primitives are interesting for at least two reasons. First, if a certain primitive exists or can be assumed to exist in nature (e.g. a noisy channel), then it can be used to build practical unconditionally-secure protocols. Second, if a primitive can be realized or approximated cryptographically (e.g. oblivious transfer), then one can construct computationally-secure cryptographic protocols with a well-defined isolated complexity assumption. The relevance of a reduction result depends on at least the following criteria:

- whether the resulting primitive is useful in applications,
- whether the assumed primitive has a natural realization in the physical world, or can efficiently be realized by cryptographic means,
- the efficiency of the reduction, for instance the number of invocations of primitive G needed to realize one occurrence of H?
- the assumption about cheating (e.g. less than n/3 cheaters), and
- which additional assumptions are made (e.g., availability of an authenticated channel between certain or all pairs of players).

Informally, a reduction of one primitive to another is secure against an adversary with some specified corruption power if the adversary can do nothing more in the protocol than what he could do in an idealized implementation of the primitive, except possibly with exponentially small probability.

Many results in cryptography can be phrased as a reduction among primitives. Some of them were mentioned above, and a few are listed below:

- Many reduction results for oblivious transfer (e.g. [8,10,14,15,16,18,20]).
- Secret-key agreement by public discussion from noisy channels as discussed in the previous section can be interpreted as the reduction of key agreement to a certain type of noisy source.
- Privacy amplification [6,5], an important sub-protocol in unconditional key agreement, can be interpreted as the reduction of key agreement to a setting in which Alice and Bob share the same string, but where Eve has some arbitrary unknown type information about the string with the only constraint being an upper bound on the total amount of information.
- Byzantine agreement protocols (e.g., see [28]) can be interpreted as the reduction of the broadcast primitive to the primitive of bilateral authenticated channels, assuming active cheating by less than n/3 of the players.
- The commitment primitive can be defined for an arbitrary number n of players, one of which commits to an input. Secret sharing can be interpreted as a reduction of such a commitment primitive to the primitive of bilateral secure communication channels, assuming only passive cheating by a non-qualified set of players. Verifiable secret sharing is like secret sharing, but security is guaranteed with respect to active cheaters (e.g. less than n/3).
- The results of Ben-Or, Goldwasser and Wigderson [3] and Chaum, Crépeau, and Damgård [12] can be interpreted as the reduction of the primitive secure multi-party computation to the primitive bilateral secure communication channels, assuming active cheating by less than n/3 of the players. If also the broadcast primitive is assumed, then less than n/2 cheaters can be tolerated [38].

4.4 General Transfer Primitives

A two-player primitive covering various types of oblivious transfer as special cases can be defined as follows: Alice inputs a random variable X, and Bob can select (by the random variable $C \in \{1, ..., n\}$) to receive any one of n random variables $Y_1, ..., Y_n$ defined by a conditional distribution $P_{Y_1,...,Y_n|X}$, such that for all (or for certain) distributions P_X Alice learns nothing about C and Bob learns nothing about X beyond what he learns from Y_C .

We refer to an (α, β) -transfer as any transfer of the described type for which for at least one distribution P_X (e.g., the uniform distribution) we have $H(X) = \alpha$ and $H(Y_i) \leq \beta$ for $1 \leq i \leq n$, and assuming the natural condition that X is determined from Y_1, \ldots, Y_n , i.e. that X contains no information that is irrelevant in the transfer. A transfer is said to hide at most γ bits if for all distributions P_X and for $1 \leq i \leq n$ we have $H(X|Y_i) \leq \gamma$.

For example, in 1-out-of-n OT of l-bit strings, X is the concatenation of the n input strings, which are Y_1, \ldots, Y_n . Such an OT is an (ln, l)-transfer and can easily be shown to hide at most (n-1)l bits. Motivated by Dodis' and Micali's [20] lower bound on reducing weak 1-out-of-N OT of L-bit strings to 1-out-of-n OT of l-bit strings we prove the following more general theorem.

Theorem 2. The reduction of any (α, β) -transfer to any transfer that hides at most γ bits requires at least $(\alpha - \beta)/\gamma$ invocations of the latter.

Proof. Let X be the input and Y be the output of the (α, β) -transfer to be realized, and let T be the entire communication taking place during the reduction protocol over the standard communication channel. Let k be the number of invocations of the second transfer, let U_1, \ldots, U_k and V_1, \ldots, V_k be the corresponding k inputs and outputs, and let $U^k = [U_1, \ldots, U_k]$ and $V^k = [V_1, \ldots, V_k]$. Then we have $H(X|V^kT) \geq \alpha - \beta$ (for at least one distribution P_X) because Bob must not learn more than β bits about X, and $H(X|U^kT) = 0$ because unless Alice enters all information about X into the protocol, she will learn something about C. We expand $H(XU^k|V^kT)$ in two different ways:

$$H(XU^{k}|V^{k}T) = H(U^{k}|V^{k}T) + H(X|U^{k}V^{k}T) = H(X|V^{k}T) + H(U^{k}|XV^{k}T),$$

and observe that $H(X|U^kV^kT) \leq H(X|U^kT) = 0$ and $H(U^k|XV^kT) \geq 0$. Applying repeatedly the chain rule and the fact the further conditioning cannot increase entropy, we obtain $\alpha - \beta \leq H(X|V^kT) \leq H(U^k|V^kT) \leq H(U^k|V^k) = \sum_{j=1}^k H(U_j|V^k, U_1 \cdots U_{j-1}) \leq \sum_{j=1}^k H(U_j|V_j) \leq k\gamma$, and the theorem follows.

5 Generalized Random Oracles

In this section we briefly sketch the definition of a new general concept, which we call generalized random oracles for lack of a perhaps better name, and describe some applications and constructions.

One motivation for introducing this concept is the fact that many proofs of the computational security of a cryptographic system (e.g. a MAC scheme or a pseudo-random permutation) based on a pseudo-random function (PRF) [25] rely on information-theoretic arguments, although this is not always made explicit in the proofs.

An adversary's attack is modeled as a usually adaptive algorithm for performing certain query operations to the system.⁷ The proof of the computational security of such a system consists of two steps: 1) a purely information-theoretic proof that the attack cannot succeed in an idealized setting where the PRF is replaced by a random function, and 2) the simple observation that if the random function is replaced by a PRF, then any efficient successful attack algorithm could be converted into an efficient distinguisher for the PRF, thus contradicting the underlying intractability assumption.

For instance, the Luby-Rackoff construction of a 2n-bit to 2n-bit pseudorandom permutation generator [29] (involving three pseudo-random functions from n bits to n bits, combined in a three-round Feistel construction) can be proved secure by showing that no adaptive algorithm querying the permutation for less than a certain (super-polynomial) number of arguments (actually $2^{n/2}$) can distinguish it from a truly random permutation with non-negligible advantage. If the three random functions are replaced by PRFs, the construction is computationally indistinguishable from a random permutation.

We sketch a general approach that allows to simplify and generalize many of the security proofs given in the literature by interpreting them as the proof of indistinguishability of two particular types of generalized random oracles. Some of the proofs can be obtained from a set of simpler information-theoretic arguments which can be formulated as results of independent interest and can serve as a toolkit for new constructions. Some of the proofs that can be revisited are those for the Luby-Rackoff construction [29] and generalizations thereof [31,36], and the analysis of the CBC MAC [1] and the XOR MAC [2].

Definition 3. A generalized random oracle (GRO) is characterized by 1) a set of query operations, each of which takes as input an argument from a certain domain and outputs a corresponding value in a certain range, and 2) a random experiment for which each elementary event in the sample space is a complete set of answers to all possible queries, with some probability distribution over the sample space.

A more operational interpretation of a GRO may help: In many cases a GRO is constructed using an array of k (usually k is exponential in a security parameter) independent random bits. The sample space hence consists of 2^k equally probable elementary events, and each access operation consists of (efficiently) evaluating a certain function involving the k bits.

The simplest form of a GRO is a random function from n bits to 1 bit (hence $k = 2^n$). The (single) query operation evaluates the random function for a given argument. A generalization is obtained by allowing other types of queries, e.g.

⁷ For example, for the case of a MAC, two query operations are allowed: evaluation of the MAC for a chosen message, and verification of a message-MAC pair (yielding a binary output) [2].

arbitrary linear combinations of the bits. Allowing outputs of general size (e.g. also n bits as in [29]) entails no essential generalization, except that a new type of binary query exists: for a given input/output pair, do they match?

The concept of locally random functions proposed in [31] also fits into this framework: these are efficient constructions of GROs with $k \ll 2^n$ and a single query operation $\{0,1\}^n \to \{0,1\}$, and which are indistinguishable from a random function for any algorithm accessing them less than, but close to k times.

Definition 4. Two GRO's \mathcal{A} and \mathcal{B} have compatible access operations if each operation for \mathcal{A} is compatible with an operation for \mathcal{B} in the sense of having the same input domain and output range. Informally, two GRO's \mathcal{A} and \mathcal{B} with compatible access operations are perfectly (statistically) indistinguishable for a given number of access operations of each type if no adaptive algorithm that can access the GROs in the specified manner has different (non-negligibly different) probability of outputting 1 when the queries are answered using \mathcal{A} or \mathcal{B} .

Note that there is no restriction on the computing power of the distinguishing algorithm; hence a GRO is an information-theoretic rather than a complexity-theoretic concept.

6 Conclusions

The three main points addressed in this paper are:

- In cryptography and more generally in computer science one generally considers only digital operations. However, all processes in the real world, including computation and communication, are physical processes involving noise and other uncertainty factors. We propose to further investigate and exploit this fact to achieve unconditional security in cryptography.
- A general definition of an information-theoretic cryptographic primitive was proposed which encompasses many primitives previously proposed in the literature and leads to new research problems on the reduction between such primitives.
- A generalized definition of a random oracle has been proposed which has applications for security proofs in complexity-theoretic cryptography.

Acknowledgments

It is impossible to list all the people with whom I have had stimulating discussions on topics related to information-theoretic cryptography. Among them, I would like to thank the following people with whom I have had the pleasure to collaborate on related topics: Charles Bennett, Gilles Brassard, Christian Cachin, Ronald Cramer, Claude Crépeau, Ivan Damgård, Matthias Fitzi, Martin Gander, Martin Hirt, Olaf Keller, and Stefan Wolf. Above all, I am grateful to Jim Massey who introduced me to information theory and cryptography. I also thank Michael Wiener and the CRYPTO 99 program committee for inviting me to give this lecture.

References

- M. Bellare, J. Kilian, and P. Rogaway, The security of the cipher block chaining message authentication code, Advances in Cryptology CRYPTO '94, Lecture Notes in Computer Science, vol. 839, Springer-Verlag, 1995.
- M. Bellare, J. Guérin, and P. Rogaway, The security of the cipher block chaining message authentication code, Advances in Cryptology CRYPTO '95, Lecture Notes in Computer Science, vol. 963, Springer-Verlag, 1994.
- 3. M. Ben-Or, S. Goldwasser, and A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pp. 1–10, 1988.
- C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, Experimental quantum cryptography, *Journal of Cryptology*, vol. 5, no. 1, pp. 3–28, Springer-Verlag, 1992.
- C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer, Generalized privacy amplification, *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 1915– 1923, 1995.
- C. H. Bennett, G. Brassard, and J.-M. Robert, Privacy amplification by public discussion, SIAM Journal on Computing, vol. 17, pp. 210–229, 1988.
- R. E. Blahut, Principles and practice of information theory, Addison-Wesley Publishing Company, 1988.
- G. Brassard and C. Crépeau, Oblivious transfer and privacy amplification, Advances in Cryptology EUROCRYPT '97, Lecture Notes in Computer Science, vol. 1233, pp. 334–345, Springer-Verlag, 1997.
- 9. C. Cachin, Entropy measures and unconditional security in cryptography, Ph. D. Thesis, ETH Zürich, Hartung-Gorre Verlag, Konstanz, 1997.
- On the foundation of oblivious transfer, Advances in Cryptology EURO-CRYPT '98, Lecture Notes in Computer Science, vol. 1403, pp. 361–374, Springer-Verlag, 1998.
- C. Cachin and U.M. Maurer, Unconditional security against memory-bounded adversaries, Advances in Cryptology - CRYPTO '97, Lecture Notes in Computer Science, vol. 1294, pp. 292–306, Springer-Verlag, 1997.
- D. Chaum, C. Crépeau, and I. Damgård, Multiparty unconditionally secure protocols, In Proc. 20th ACM Symposium on the Theory of Computing (STOC), pages 11–19, 1988.
- 13. T. M. Cover and J. A. Thomas, *Elements of information theory*, Wiley Series in Telecommunications, 1992.
- C. Crépeau, Equivalence between two flavours of oblivious transfer, Advances in Cryptology - CRYPTO '87, Lecture Notes in Computer Science, pp. 350–354, Springer-Verlag, 1988.
- Efficient cryptographic protocols based on noisy channels, Advances in Cryptology EUROCRYPT '97, Lecture Notes in Computer Science, vol. 1233, pp. 306–317, Springer-Verlag, 1997.
- C. Crépeau and J. Kilian, Achieving oblivious transfer using weakened security assumptions, 29th Symposium on Foundations of Computer Science, pp. 42–52, IEEE, 1988.
- 17. I. Csiszár and J. Körner, Broadcast channels with confidential messages, *IEEE Transactions on Information Theory*, vol. IT-24, pp. 339–348, 1978.
- I. Damgård, J. Kilian, and L. Salvail, On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions, Advances in Cryptology EUROCRYPT '99, Lecture Notes in Computer Science, vol. 1592, pp. 56-73, Springer-Verlag, 1999.

- 19. W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.
- Y. Dodis and S. Micali, Lower bounds for oblivious transfer reductions, Advances in Cryptology - EUROCRYPT '99, Lecture Notes in Computer Science, vol. 1592, pp. 42–55, Springer-Verlag, 1999.
- W. Feller, An introduction to probability theory and its applications, 3rd edition, vol. 1, Wiley International, 1968.
- M. J. Fischer and R. N. Wright, Bounds on secret key exchange using a random deal of cards, *Journal of Cryptology*, vol. 9, no. 2, pp. 71–99, Springer-Verlag, 1996.
- P. Gemmell and M. Naor, Codes for interactive authentication, Advances in Cryptology CRYPTO '93, Lecture Notes in Computer Science, vol. 773, pp. 355–367, Springer-Verlag, 1993.
- 24. E. N. Gilbert, F. J. MacWilliams, and N. J. A. Sloane, Codes which detect deception, *Bell Syst. Tech. J.*, vol. 53, No. 3, 1974, pp. 405–424.
- 25. O. Goldreich, S. Goldwasser, and S. Micali, How to construct random functions, *Journal of the ACM*, vol. 33, no. 4, pp. 210–217, 1986.
- O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symposium on the Theory of Computing (STOC)*, pp. 218–229, 1987.
- J. Håstad, R. Impagliazzo, L. Levin, and M. Luby, Construction of a pseudorandom generator from any one-way function, Technical Report no. 91-068, ICSI, Berkeley, CA, 1991.
- 28. L. Lamport, R. Shostak, and M. Pease, The Byzantine generals problem, *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382–401, 1982.
- M. Luby and C. Rackoff, How to construct pseudorandom permutations from pseudorandom functions, SIAM Journal on Computing, vol. 17, no. 2, pp. 373–386, 1988.
- U. M. Maurer, Conditionally-perfect secrecy and a provably-secure randomized cipher, *Journal of Cryptology*, vol. 5, pp. 53–66, Springer-Verlag, 1992.
- 31. —, A simplified and generalized treatment of Luby-Rackoff pseudo-random permutation generators, *Advances in Cryptology EUROCRYPT '92*, Lecture Notes in Computer Science, vol. 658, pp. 239–255, Springer-Verlag, 1992.
- 32. —, Secret key agreement by public discussion from common information, *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 733–742, 1993.
- 33. —, A unified and generalized treatment of authentication theory, *Proceedings* 13th Symp. on Theoretical Aspects of Computer Science STACS '96, Lecture Notes in Computer Science, vol. 1046, pp. 387–398, Springer-Verlag, 1996.
- 34. —, Information-theoretically secure secret-key agreement by NOT authenticated public discussion, *Advances in Cryptology EUROCRYPT '97*, Lecture Notes in Computer Science, vol. 1233, pp. 209–225, Springer-Verlag, 1997.
- 35. U. M. Maurer and S. Wolf, Unconditionally secure key agreement and the intrinsic conditional information, *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 499–514, 1999.
- 36. M. Naor and O. Reingold, On the construction of pseudorandom permutations: Luby-Rackoff revisited, *Journal of Cryptology*, vol. 12, no. 1, pp. 29–66, 1999.
- 37. M.O. Rabin, personal communication, 1998.
- 38. T. Rabin and M. Ben-Or, Verifiable secret sharing and multiparty protocols with honest majority, *Proc. 21st ACM Symposium on the Theory of Computing (STOC)*, pp. 73–85, 1989.
- 39. C. E. Shannon, Communication theory of secrecy systems, *Bell System Technical Journal*, vol. 28, pp. 656–715, 1949.
- 40. —, A mathematical theory of communication, *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, 1948.

- 41. G. J. Simmons, A survey of information authentication, *Proceedings of the IEEE*, vol. 76, pp. 603–620, 1988.
- D. R. Stinson, Universal hashing and authentication codes, Advances in Cryptology

 CRYPTO '91, Lecture Notes in Computer Science, vol. 576, pp. 74–85, Springer-Verlag, 1992.
- 43. G. S. Vernam, Cipher printing telegraph systems for secret wire and radio telegraphic communications, *Journal of the American Institute for Electrical Engi*neers, vol. 55, pp. 109–115, 1926.
- M. N. Wegman and J. L. Carter, New hash functions and their use in authentication and set equality, *Journal of Computer and System Sciences*, vol. 22, pp. 265-279, 1981.
- 45. S. Wolf, Strong security against active attacks in information-theoretic secret-key agreement, *Advances in Cryptology ASIACRYPT '98*, Lecture Notes in Computer Science, vol. 1514, pp. 405–419, Springer-Verlag, 1998.
- Unconditional security in cryptography, Proceedings of Summer School in Cryptology and Data Security, July 1998, Aarhus, Denmark, Lecture Notes in Computer Science, vol. 1561, pp. 217–250, Springer-Verlag, 1999.
- 47. —, Information-theoretically and unconditionally secure key agreement in cryptography, Ph.D. Thesis no. 13138, ETH Zurich, 1999.
- 48. A. D. Wyner, The wire-tap channel, *Bell System Technical Journal*, vol. 54, no. 8, pp. 1355–1387, 1975.

Information Theoretically Secure Communication in the Limited Storage Space Model

Yonatan Aumann 1 and Michael O. Rabin 2,3*

Department of Computer Science, Bar-Ilan University, Ramat Gan 52900, Israel aumann@cs.biu.ac.il

Department of Computer Science, The Hebrew University of Jerusalem Jerusalem 91904, Israel

rabin@cs.huji.ac.il

DEAS, Harvard University, Cambridge, MA 02138, USA
rabin@deas.harvard.edu

Abstract. We provide a simple secret-key two-party secure communication scheme, which is provably information-theoretically secure in the limited-storage-space model. The limited-storage-space model postulates an eavesdropper who can execute arbitrarily complex computations, and is only limited in the total amount of storage space (not computation space) available to him. The bound on the storage space can be arbitrarily large (e.g. terabytes), as long as it is fixed. Given this bound, the protocol guarantees that the probability of the eavesdropper of gaining any information on the message is exponentially small. The proof of our main results utilizes a novel combination of linear algebra and Kolmogorov complexity considerations.

1 Introduction

The most basic problem in cryptography is that of communication over an insecure channel, where a Sender S wishes to communicate with a Receiver, R, while an Eavesdropper, E, is tapping the line. To achieve privacy, the Sender and Receiver may share a common key. In a seminal work, Shannon [10] proved that if the eavesdropper has complete access to the communication line, and is not bounded in any way, then perfect, information theoretically secure communication is only possible if the entropy of the key space is at least as large as that of the Plaintext space. In essence, this means that if the eavesdropper is unbounded then the one-time-pad scheme, where the size of the secretly shared pad equals the size of the message, is the best possible scheme. This, of course, is impractical for most applications. Thus, to obtain practical solutions one must place some bounds on the eavesdropper's power. Most of modern cryptography has proceeded along the line of assuming that the eavesdropper is computationally bounded and devising schemes that are computationally hard to break.

^{*} Research supported, in part, by NSF Grant NSF-CCR-97-00365, at Harvard University.

These results, though unquestionably impressive, suffer from two drawbacks. First, they are based on unproven complexity assumptions. Second, many of the methods tend to require considerable computations, even from the Sender and Receiver. In theory, any polynomial computation is considered feasible for the participants. In practice, some of the polynomials are so large as to render the solution impractical. In other cases, even when the solutions are practical (e.g. RSA), the amount of computation limits their use to high-security applications or key-exchange sessions.

The Limited Storage Space Model. Recently, there has been development in constructing secure communication protocols not based on computational complexity. In this paper we consider the *limited storage space* model, where the security guarantees are based on the limited storage space available to the eavesdropper. This model, first introduced by Maurer [7], assumes that there is a known bound, possibly very large - but fixed, on the amount of storage-space available to both the Sender and Receiver, and a (possibly much larger but fixed) bound on the storage space available to the Eavesdropper. It is important to differentiate between this model and the well-known so called bounded space model (e.g. log-space). The bounded space model considers cases where the space is usually very limited, and serves as *computing* space. In this case, the space-bound is in effect a limitation on computational power. The limited storage space model, by contrast, allows very large amounts of storage space, placing the limit only to practically feasible capacities (e.g. a terabyte or several terabytes) of storage. At the same time we do not stipulate any limitations on the computational power of the eavesdropper who is trying to subvert the secrecy of the protocol. Given the bound on the eavesdropper's storage space, the model enables to obtain results which are *information theoretic*, not depending on any unproven assumptions. Furthermore, the model enables us to construct a very simple and efficient protocol, requiring very little computation and storage space for the Sender and Receiver. Informally, the limited storage space model assumes a publicly accessible source of random bits, such as a high rate broadcast of a string α of random bits, equally accessible to the Sender, Receiver and Eavesdropper. Let the length of α be $|\alpha| = nm$, where m is the length of the message to be securely sent. It is assumed that the Eavesdropper is limited to storing E < n bits, say E = n/5. The Eavesdropper can listen to the whole of α and compute and store any function $f(\alpha)$, provided that $|f(\alpha)| \leq E$. In one version, the model postulates that the sender and receiver share a secret key s where |s| = O(logn). Using s and listening to α , the sender and receiver both read and store ℓ chosen locations of α and compute from those bits a one-time pad X, |X| = m. The pad X is used by the Sender to encrypt a message M, |M| = m. We show that the pad X can be used as a secure one-time-pad for secret communication between the Sender and the Receiver.

The Limited Storage Space Model - Previous Work. In a ground-breaking work, Maurer [7] presented the first protocol for private-key secure communication in the Limited-Storage-Space model described above. However, in [7], the proof of

the security of the protocol is provided only for the case where the bound on the eavesdropper is not only on the *space* available to her, but also on the total number of random bits she can access. In particular, it is assumed that the eavesdropper can only access a constant fraction of the random bits of α . The analysis of [7] does not provide a proof for the general limited-storage-space case, where the eavesdropper can access all the bits of α , can perform any computation on these bits, and store the results of the computation in the bounded space available to him. It was left as an open question in [7] if any security result can be proved using the limited-storage-space assumption alone. Recently, Cachin and Maurer [4] provided a protocol for which they prove security based solely on the limited-storage-space assumption. However, this protocol is considerably more complex than the original protocol, employing advanced privacy amplification techniques. Their proof uses sophisticated Renyi entropy considerations. To assure that the probability of revelation to the eavesdropper of the secret message M be smaller than ϵ , the protocol of [4] requires the Sender and Receiver to store $\ell \log n$ and transmit ℓ bits, where $\ell = 3/\epsilon^2$. Thus if we prudently require $\epsilon = 10^{-6}$, we get that $\ell = 3 \cdot 10^{12}$; the length n of the random string α can be 2^{40} , so that $\log n = 40$. Thus for this choice of ϵ , the Sender and Receiver have to store and transmit very large numbers of bits. Furthermore, the protocol calls for a multiplication operation in a large field of ℓ -bit numbers.

Our Results. In this paper we return to the simple original protocol of [7], and show that security can be proved, for a slightly modified protocol, based only on the limited storage space assumption. Altogether, we obtain a secure secret key communication scheme secure against any eavesdropper with limited storage space. The protocol is very simple for the Sender and Receiver, necessitating only the elementary XOR operations, and minimal storage space. Specifically, the secret shared key s has $k \log n$ bits, where k is a security parameter. For a secret transmission of a message M of length m, the Sender and Receiver have each to read from α and store just km bits and the one-time pad X is computed from those bits, like in [7], by just XOR operations. Finally, the probability of the adversary Eavesdropper to gain even one-bit information on the message M, is smaller than $2^{-k/5}$, i.e. exponentially small in k. The results are obtained using novel techniques, employing linear algebra and Kolmogorov complexity arguments. An exact formulation of the results is provided in Section 2. We note, however, that our protocol requires a longer random string α , than does that of [7]. It remains a open problem to reduce this number. It will also be interesting to see whether the new methods can considerably improve the constants in the protocol of [4], or some variation of this protocol.

Related Work. In a series of papers, Maurer and colleagues [7,8,2,4] consider secure communication solutions not based on computation complexity. In [8,2] the authors consider the setting where all parties (honest and eavesdropper) communicate over noisy channels. The model of limited-storage-space was first introduced in [7] and further developed in [4], as discussed above. The limited

```
Communication Protocol.

Message M = (M_1, ..., M_m) \in \{0, 1\}^m. Secret key s = (\sigma_1, ..., \sigma_k) \in \{1, ..., n\}^k

1 for i = 1 to m do

2 for j = 1 to n do

3 Broadcast random \alpha_j^{(i)} (either produced by \mathcal{S}, \mathcal{R} or an outside source).

4 If j \in s then

5 \mathcal{R} and \mathcal{S} store \alpha_j^{(i)} in memory end for loop

6 \mathcal{S} and \mathcal{R} set X_i := \bigoplus_{j=1}^k \alpha_{\sigma_j}^{(i)} end for loop

7 \mathcal{S} and \mathcal{R} set X = (X_1, ..., X_m)

8 \mathcal{S} computes Y = X \oplus M. Sends Y to \mathcal{R}.

9 \mathcal{R} decryptes M = X \oplus Y
```

Fig. 1. Communication Protocol

storage space model in the context of Zero-Knowledge proofs was first studied by De Santis, Persiano and Yung [5], and then by Aumann and Feige [1].

2 The Protocol

We consider a secret key setting, where the sender and receiver share a small secret key. Sender S wants to send a message $M \in \{0,1\}^m$ to the receiver R over an insecure channel. An eavesdropper E may be tapping the communication line between S and R. We assume that there is a known bound, E, on the total storage space available to E. Let k be a security parameter, and let n = 5E. Given a private key of size $k \log n$ chosen uniformly at random, our scheme guarantees information-theoretic secrecy with probability $\geq 1 - 2^{k/5}$.

The scheme is essentially that of [7] (with one exception, which is discussed in the end of this section). First, S and R produce a shared "one-time-pad", $X = (X_1, \ldots, X_m)$. Then S computes $Y = X \oplus M$. He then sends Y to R, who then computes $M = X \oplus Y$ to obtain the original message.

To produce each shared bit, X_i , i = 1, ..., m, the protocol employs a long random "noise" string $\alpha^{(i)}$, of size n = 5E, broadcasted from \mathcal{S} to \mathcal{R} . Alternatively, the string $\alpha^{(i)}$ may be available to both and \mathcal{R} from an outside source, e.g. random channel noise, or a special public "noise" broadcast. We assume that \mathcal{E} has full access to $\alpha^{(i)}$ while it is being broadcast. In particular, \mathcal{E} can access all the bits of $\alpha^{(i)}$ and perform on them any possible computation, polynomial or non-polynomial. The sole restriction is that the total space available to \mathcal{E} , for storing the output of her computation on $\alpha^{(i)}$ is bounded by E. Let s be the secret key. We interpret s as a sequence $\sigma_1, \ldots, \sigma_k$ of integers in [1, n]. As the random string $\alpha^{(i)}$ is broadcasted, both the \mathcal{E} and the \mathcal{R} retain the k bits $\alpha^{(i)}_{\sigma_1}, \ldots, \alpha^{(i)}_{\sigma_k}$. Both players then compute $X_i = \bigoplus_{j=1}^k \alpha^{(i)}_{\sigma_j}$, to produce the i-th random bit of the one-time-pad X. A detailed description of the protocol is provided in Figure 1.

We prove that the bit X_i can be used as a secure one-time-pad bit for communication between \mathcal{S} and \mathcal{R} . In particular, we show that the probability of \mathcal{E}

computing X_i correctly is $\leq 1/2 + 2^{-k/5}$. This holds true even if s is revealed to \mathcal{E} after his computation on the string $\alpha^{(i)}$ is completed. Note that for the honest players the scheme is very simple, requiring only $k \log n$ storage space and XOR computations.

To understand the protocol, consider the case where \mathcal{E} can only store original bits of α , without performing any computations. The storage space available to \mathcal{E} allows her to store at most one fifth of the bits of α . Thus, for any i, the probability that all bits $\alpha_{\sigma_i}^{(i)}$ are available to \mathcal{E} is exactly 5^{-k} . If \mathcal{E} happens to have stored exactly the right bits, then she knows the value of X_i . Otherwise, it can be shown that \mathcal{E} has no information on X_i , and can only guess its value with probability 1/2. Thus, if \mathcal{E} can only store original bits of α then her probability of guessing X_i is $1/2 + 5^{-k}$. A result in this spirit was proven by Maurer [7]. In our model, however, we allow \mathcal{E} to perform any computation on the bits of α , and retain any function of these bits, provided that the total size of the output does not exceed \mathcal{E} bits. We prove that the extra computations can give \mathcal{E} at most a very marginal advantage. We prove that in any case, the probability of \mathcal{E} of correctly computing even one bit of knowledge about X is at most $1/2 + 2^{-k/5}$. The bound is information theoretic and does not depend on any unproven complexity assumptions.

The Main Theorem. We prove a strong statement concerning the security of the scheme. We show that even if the Eavesdropper is provided with the full key s after α is broadcasted, she still has negligible probability of gaining any information on the message M. Furthermore, we apply the storage restriction only at one point in time - namely, the time immediately after α is transmitted. Thus, the dynamics can be described as follows:

Phase I:

- (a) The stream α is generated and transmitted $(\alpha \in \{0,1\}^{nm})$.
- (b) The Eavesdropper can perform any computation on α , with no restrictions on space or time.

Following this phase, the eavesdropper can store n/5 bits. We denote by η , $|\eta| = n/5$, the information stored after this phase.

Phase II: The Eavesdropper is provided with $Y = X \oplus M$ and the key s. Based on η , Y and s, the Eavesdropper tries to gain information on M.

Thus, any algorithm, A of the eavesdropper, is actually a pair of algorithms $A = (A_1, A_2)$, where A_1 is the algorithm for the first phase and A_2 is the algorithm for the second phase (after \mathcal{E} receives s). The first algorithm gets α as an input, and outputs $\eta = A_1(\alpha)$, with $|\eta| = n/5$. The second algorithm A_2 gets s, η and Y as inputs, and outputs a single bit, $\delta = A_2(\eta, s, Y)$. Accordingly, we denote the entire eavesdropper's algorithm as $A(\alpha, s, Y)$.

We prove that, for any message M, sent from S to R, the probability of E to gain any additional information on M from the protocol is exponentially small (in k). Specifically, consider any two possible distributions on messages, $D^{(0)}$ and $D^{(1)}$. The Eavesdropper wishes to know from which of the two distribution

the message was drawn. For a message M and an eavesdropper's algorithm $A = (A_1, A_2)$, we denote by A(M) the entire output of the algorithm for a message M, i.e. $A(M) = A_2(A_1(\alpha), s, X(s, \alpha) \oplus M)$, with α and s chosen uniformly at random.

Theorem 1. For n large enough, for any distributions $D^{(0)}, D^{(1)}$, and any Eavesdropper's algorithm A, which uses at most E = n/5 storage space,

$$\left| \Pr \left[A(M) = 1 | M \in D^{(1)} \right] - \Pr \left[A(M) = 1 | M \in D^{(0)} \right] \right| \le 2^{-k/5},$$

where the probability is taken over the random choices of α , the random secret key s, and the random choices of M from the distributions $D^{(0)}$ and $D^{(1)}$.

In particular the theorem says that if there are only two possible messages, $M^{(0)}$ and $M^{(1)}$, and the Eavesdropper wishes to know which message was sent, then she has only an exponentially small probability of success. Note that there is no limit on the time complexity of the eavesdropper's algorithm A. The only limit is that the storage is bounded by E = n/5. Also note that the result is non-uniform, in the sense that the algorithm A may be tailored to the specific distributions $D^{(0)}$ and $D^{(1)}$.

We note that, in addition to providing provable secrecy, our scheme provides two important security features, not usually provided in complexity-based schemes. First, as noted, the secrecy is guaranteed even if following the transmission the secret-key is fully revealed. Thus, the system is secure against future leakage of the key. Secondly, the system is also secure against the event in which the eavesdropper \mathcal{E} subsequently obtains more storage space. The bound we have on \mathcal{E} 's storage space need only be true for her current available storage. Any future additional storage will not give her any advantage. Thus, future advances in storage technology do not threaten the secrecy of current communications. This is in contrast to most-all complexity-based schemes, were messages can be stored now and deciphered later, if and when computing technology allows (e.g. using quantum computers to factor numbers).

How many random bits are necessary? The protocol as described above uses n random bits for each message bit, for a total of nm random bits. The [7] protocol, in contrast, uses only n bits in total, for the entire message. This is achieved in the following way. A single string α of length n is broadcast. The bit X_1 is defined as in our protocol. For the subsequent bits, X_i is defined to be $X_i := \bigoplus_{j=1}^k \alpha_{s_{j+i-1}}$. Thus, all the X_i 's are obtained from a single α of length n. For the model of [7], where the eavesdropper can only access E = n/5 bits, Maurer proves that the reduced bit protocol suffices to guarantee security. The proof, however, does not carry over to our setting, where the eavesdropper can access all the bits and the sole bound is on the space available to the eavesdropper. Our proof, as described in the next section, necessitates n random bits for each message bit. It remains an important open problem to extend the proof to allow for a similar number of bits as in the original [7] protocol.

3 The Proof

We provide the proof in several stages. First, we prove that it is sufficient to prove the theorem for the case where there are only two possible messages. Next, we prove the theorem for the case of one-bit messages. We do this by proving that if for a given α , the knowledge of η ($|\eta| = n/5$) helps the eavesdropper in reconstructing the one-bit message M for many different secret-keys s, then the Kolmogorov Complexity of α must be small. Since most α 's have high Kolmogorov complexity, this shows that the eavesdropper's probability of being correct is small. Next, having proven the theorem for single bit messages, we consider the case of long messages that differ in a single bit. Finally, we prove the full theorem.

Notations: For a string $\alpha^{(i)} = (\alpha_1^{(i)}, \dots, \alpha_n^{(i)}), (\alpha^{(i)} \in \{0, 1\}^n)$, and $s = (\sigma_1, \dots, \sigma_k)$ we denote $s(\alpha^{(i)}) = \bigoplus_{j=1}^k \alpha_{\sigma_j}^{(i)}$. For $\alpha = (\alpha^{(1)}, \dots, \alpha^{(m)})$, we denote $s(\alpha) = (s(\alpha^{(1)}), \dots, s(\alpha^{(m)}))$. We also denote $X(s, \alpha) = s(\alpha)$.

3.1 From Distributions to Messages.

Lemma 1. Theorem 1 holds iff for n large enough, for any two messages $M^{(0)}$ and $M^{(1)}$ and any Eavesdropper's algorithm A, which uses at most n/5 storage space,

$$\left| \Pr \left[A(M^{(1)}) = 1 \right] - \Pr \left[A(M^{(0)}) = 1 \right] \right| \le 2^{-k/5},$$

where the probability is taken over the random choices of α and the secret key s.

Proof. Clearly, if Theorem 1 holds, then in particular it holds when the distribution $D^{(1)}$ is concentrated solely on $M^{(1)}$ and $D^{(0)}$ solely on $M^{(0)}$.

Conversely, suppose that

$$\left| \Pr \left[A(M^{(1)}) = 1 \right] - \Pr \left[A(M^{(0)}) = 1 \right] \right| \le 2^{-k/5},$$

for any two messages. Let $D^{(0)}$ and $D^{(1)}$ be two distributions. W.l.o.g. assume that

$$\Pr\left[A(M) = 1 | M \in D^{(1)}\right] - \Pr\left[A(M) = 1 | M \in D^{(0)}\right] \ge 0.$$

Let $M^{(1)}$ be the message such that $\Pr\left[A(M^{(1)})=1\right]$ is the largest, and let $M^{(0)}$ be the message such that $\Pr\left[A(M^{(0)})=1\right]$ is the smallest. Then,

$$\begin{split} \Pr\left[A(M) = 1 | M \in D^{(1)}\right] - \Pr\left[A(M) = 1 | M \in D^{(0)}\right] = \\ \sum_{M} \Pr\left[A(M) = 1\right] \Pr_{D^{(1)}}\left[M\right] - \sum_{M} \Pr\left[A(M) = 1\right] \Pr_{D^{(0)}}\left[M\right] \le \\ \left|\Pr\left[A(M^{(1)}) = 1\right] - \Pr\left[A(M^{(0)}) = 1\right]\right| \le 2^{-k/5}. \end{split}$$

Thus, it is sufficient to focus on the case of just two possible messages.

3.2 Single Bit Secrecy

We now prove the theorem for the case of a single bit message, i.e. m=1. We use the following notations. Since m=1, we have $\alpha=\alpha^{(1)}$. Thus, we omit the superscript from α and write $\alpha=(\alpha_1,\ldots,\alpha_n)$ $(\alpha_j\in\{0,1\})$. Similarly, we denote $X=X_1$. Let $K=n^k$ and $N=2^n$. Let $S=(s_1,s_2,\ldots,s_K)$ be an enumeration of all possible secret keys, and $\mathcal{A}=(\alpha_1,\ldots,\alpha_N)$ be an enumeration of all strings of length n. For a bit b we denote $\bar{b}=(-1)^b$ (i.e. we replace 1 by -1 and 0 by 1). For a sequence $B=(b_1,\ldots,b_K)$ we denote $\bar{B}=(\bar{b}_1,\ldots,\bar{b}_K)$. For a sequence α we denote $v(\alpha)=(s_1(\alpha),\ldots,s_K(\alpha))$.

Preliminaries. For $v \in \{1, -1\}^K$ define the discrepancy of v as $d(v) = |\sum_{i=1}^K v_i|$.

Lemma 2. Let $\alpha \in \{0,1\}^n$ be such that the fraction of 1's and the fraction of 0's in α is no less then 1/8, then

$$d(v(\alpha)) < \frac{K}{2^{0.4k}}.$$

Proof. Assume k is odd. Let p be the fraction of 1's in α . Set q=1-p. Consider a random choice of $s=(s_1,\ldots,s_k)\in S$. Since k is odd,

$$s(\alpha) = 1 \Leftrightarrow |\{i : \alpha_{s_i} = 1\}| \text{ is odd }$$

For any $0 \le t \le k$,

$$\Pr\left[\left|\left\{i:\alpha_{s_i}=1\right\}\right|=t\right]=\binom{k}{t}\,p^tq^{k-t}.$$

Thus,

$$\Pr[s(\alpha) = 1] = \sum_{t \text{ odd}} {k \choose t} p^t q^{k-t}.$$
 (1)

Now,

$$1 = (p+q)^k = \sum_{t} \binom{k}{t} p^t q^{k-t} \tag{2}$$

$$(p-q)^k = \sum_t \binom{k}{t} (-1)^{k-t} p^t q^{k-t} \tag{3}$$

Since k is odd, when t is even, $(-1)^{k-t} = -1$. Thus, adding (2) and (3),

$$\frac{1 + (p-q)^k}{2} = \sum_{t \text{ odd}} \binom{k}{t} p^t q^{k-t}.$$

Together with (1), we get,

$$\frac{1}{2} - \frac{1}{2^{0.4k+1}} < \Pr\left[s(\alpha) = 1\right] = \frac{1 + (p-q)^k}{2} < \frac{1}{2} + \frac{1}{2^{0.4k+1}}.$$

For k even, an analogous argument works by considering $\Pr[s(\alpha) = 0]$ and the number of zeros in α .

Thus,

$$D(v(\alpha)) = K |\Pr\left[s(\alpha) = 1\right] - \Pr\left[s(\alpha) = \right]| < \frac{K}{2^{0.4k}}.$$

Let

$$\mathcal{D} = \left\{ \alpha \in \mathcal{A} : d(v(\alpha)) > \frac{K}{2^{0.4k}} \right\} \tag{4}$$

be the set of vectors α with a large discrepancy.

Corollary 3 For $c \geq 0.798$, $|\mathcal{D}| \leq 2^{cn}$.

Proof. For a string α denote by $z(\alpha)$ the number of zeros in α . By Lemma 2, $\alpha \in \mathcal{D}$ only if $z(\alpha) < n/8$ or $z(\alpha) > 7n/8$. For a random α , $E(z(\alpha)) = n/2$. Thus, by the Chernoff bound ([9] p. 70, Theorem 4.2),

$$\Pr\left[z(\alpha) < n/8 \text{ or } z(\alpha) > 7n/8\right] = 2\Pr\left[z(\alpha) < \frac{n}{8}\right] \le 2\Pr\left[z(\alpha) < \left(1 - \frac{3}{4}\right)E(z(\alpha))\right] \le 2\exp\left(-\frac{n}{2}\left(\frac{3}{4}\right)^2\frac{1}{2}\right) < 2^{-0.202n}.$$

for n sufficiently large.

Lemma 4. Let $\alpha, \beta \in \mathcal{A}$,

$$v(\alpha) \otimes v(\beta) = v(\alpha \oplus \beta),$$

where \otimes is the coordinate-wise multiplication.

Proof. For each $s \in S$

$$\overline{s(\alpha)} \cdot \overline{s(\beta)} = (-1)^{s(\alpha)} \cdot (-1)^{s(\beta)} = (-1)^{(s(\alpha)) \oplus (s(\beta))} = (-1)^{s(\alpha \oplus \beta)} = \overline{s(\alpha \oplus \beta)}$$

Single Bit Case. For the case of single bit messages the only two possible messages are M=0 and M=1. For a given Y, M=1 iff X=1-Y. Thus, in order for the Eavesdropper to distinguish between M=1 and M=0, she must be able to distinguish between X=0 and X=1. Consider an algorithm $A=(A_1,A_2)$ of the Eavesdropper for guessing $X=s(\alpha)$. Given α , let $\eta=A_1(\alpha)$ be the information stored by the Eavesdropper following the first phase. By definition $|\eta|=n/5$. Next, when provided with s, and s, algorithm s0 outputs a bit s1, in hope that s2, when s3 it only needs to guess s4. For a message s5, we denote s6, we denote s6, where s6 and s6 are chosen at random.

Definition 1 We say that A_2 is good for α if there exists an $\eta \in \{0,1\}^{n/5}$ such that $\Pr[A_2(\eta,s)=s(\alpha)] \geq \frac{1}{2} + \frac{1}{2^{0.4k/2}}$, where the probability is taken over the random choices of s.

We prove that for any A_2 , for almost all α 's, A_2 is not good.

Let us concentrate on a given η . For the given η , let

$$B = B_{\eta} = (A_2(\eta, s_1), \dots, A_2(\eta, s_K))$$

(where s_1, \ldots, s_K is the enumeration of all possible keys). The vector B is an enumeration of the answers of A_2 on input η , given the various keys s_i . For a given answer vector B, let

$$L_B = \left\{ \alpha : |\bar{B} \cdot v(\alpha)| \ge \frac{2K}{2^{0.4k/2}} \right\}.$$

 L_B is the set of α 's for which the answers in B are good. By definition, if A_2 is good for α then $\alpha \in L_{B_{\eta}}$, for some η . We now bound the size of L_B , for any B. Let V be the $K \times N$ matrix, whose columns are $v(\alpha_1), \ldots, v(\alpha_N)$, i.e.

$$V = \begin{pmatrix} \overline{s_1(\alpha_1)} & \overline{s_1(\alpha_2)} & \cdots & \overline{s_1(\alpha_N)} \\ \overline{s_2(\alpha_1)} & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \overline{s_K(\alpha_1)} & \cdots & \cdots & \overline{s_K(\alpha_N)} \end{pmatrix}$$

Consider a specific $B \in \{0,1\}^K$. Let $L^+ = \{\alpha : \bar{B} \cdot v(\alpha) \ge \frac{2K}{2^{0.4k/2}}\}$ and $L^- = L_B - L^+$. We bound the size of L^+ . The proof for L^- is analogous.

Let 1_{L^+} be the characteristic vector of L^+ $(1_{L^+} \in \{0,1\}^N)$. For any $i, \bar{B} \cdot V \cdot e_i = \bar{B} \cdot v(\alpha_i)$ (where e_i is the unit vector with 1 in the *i*-th coordinate). Thus, by definition of L^+ ,

$$\bar{B} \cdot V \cdot 1_{L^{+}} \ge |L^{+}| \cdot \frac{2K}{2^{0.4k/2}}.$$
 (5)

On the other hand, by the Cauchy-Schwartz inequality,

$$\bar{B} \cdot V \cdot 1_{L^{+}} \le \|\bar{B}\| \cdot \|V \cdot 1_{L^{+}}\|.$$
 (6)

Since $\bar{B} \in \{1, -1\}^K$, we have

$$\|\bar{B}\| = \sqrt{K}.\tag{7}$$

Next, by definition,

$$\|V \cdot 1_{L^+}\|^2 = 1_{L^+}^T \cdot V^T V \cdot 1_{L^+}.$$

Consider the matrix $H = V^T V$. Set $H = (h_{i,j})$. By definition and Lemma 4, $|h_{i,j}| = d(v(\alpha_i) \otimes v(\alpha_j)) = d(v(\alpha_i \oplus \alpha_j))$. Thus,

$$|h_{i,j}| \le \begin{cases} K & \alpha_i \oplus \alpha_j \in \mathcal{D} \\ \frac{K}{2^{0.4k}} & \text{otherwise} \end{cases}$$

(where \mathcal{D} is the set of α 's with large discrepancy (eq. (4)). Let

$$\delta(i) = \begin{cases} 1 & \alpha_i \in L^+ \\ 0 & \alpha_i \not\in L^+ \end{cases}$$

We have,

$$||V \cdot 1_{L^{+}}|| = 1_{L^{+}}^{T} \cdot V^{T} V \cdot 1_{L^{+}} = \sum_{i=1}^{N} \sum_{j=1}^{N} |h_{i,j}| \delta(i) \delta(j) \leq$$

$$\sum_{i=1}^{N} \delta(i) \left(\sum_{j:\alpha_{i} \oplus \alpha_{j} \in \mathcal{D}} |h_{i,j}| + \sum_{j:\alpha_{i} \oplus \alpha_{j} \notin \mathcal{D}} |h_{i,j}| \delta(j) \right) \leq$$

$$\sum_{i:\alpha_{i} \in L^{+}} \left(\sum_{j:\alpha_{i} \oplus \alpha_{j} \in \mathcal{D}} |h_{i,j}| + \sum_{j:\alpha_{j} \in L^{+},\alpha_{i} \oplus \alpha_{j} \notin \mathcal{D}} |h_{i,j}| \right) \leq$$

$$|L^{+}| \left(2^{cn} K + |L^{+}| \frac{K}{2^{0.4k}} \right)$$
(8)

(for c as in corollary 3). Combining Equations (5), (6), (7), and (8), we get,

$$|L^{+}| \cdot \frac{2K}{2^{0.4k/2}} \le \sqrt{K} |L^{+}|^{1/2} \left(2^{cn}K + \frac{|L^{+}|K}{2^{0.4k}}\right)^{1/2}$$

Solving for $|L^+|$ gives,

$$3|L^+| < 2^{cn+0.4k}$$

Similarly, for $L^-, 3|L^-| \le 2^{cn+0.4k}$.

In all we conclude,

Lemma 5. For any possible answer vector B, $|L_B| \leq 2^{cn+0.4k}$.

We now relate the Kolmogorov complexity of α to the success probability of A_2 on α (see [6] for a comprehensive introduction to Kolmogorov Complexity). For a string α denote by $C(\alpha|A_2)$ the Kolmogorov complexity of α , with regards to a description of A_2 .

Corollary 6 For any algorithm A_2 and $\alpha \in \{0,1\}^n$ if A_2 is good for α , then

$$C(\alpha|A_2) \le \frac{n}{5} + cn + 0.4k.$$

Proof. If A_2 is good then, given A_2 , the string α can be fully specified by:

- 1. $\eta n/5$ bits, and
- 2. the index i_{α} of α in $L_{B_{\eta}}$ (in lexicographic order) at most cn + 0.4k bits,

Given this information, α is constructed as follows. First, using A_2 and η , the answer vector $B_{\eta} = (A_2(\eta, s_1), \dots, A_2(\eta, s_K))$ is constructed. Next, all possible strings $\beta \in \{0, 1\}^n$ are constructed one by one, in lexicographic order. For each

string, β , we check if $|\bar{B} \cdot v(\beta)| \ge \frac{2K}{2^{0.4k/2}}$. If so, it is marked as a member of L_B . The desired string α is the i_{α} member of L_B .

Since c < 0.8, we have $\frac{n}{5} + cn + 0.4k < n$ for n sufficiently large. Thus, only a negligible fraction $(2^{-\Omega(n)})$ of the α 's have $C(\alpha|A_2) \leq \frac{n}{5} + cn + 0.4k$. Thus, since α is chosen at random, there is only a negligible probability that A_2 is good for α , even for an optimal A_1 .

3.3 Multi-bit Secrecy - The One-more Bit Problem

Next, we consider the secrecy of the "one-time-pad" X for the case m > 1. Thus, we consider the case where $X = (X_1, \ldots, X_m)$. We consider the following problem. Suppose that the Eavesdropper is given all of X except for the last bit, $X_m = s(\alpha^{(m)})$. Her aim is to guess the value of X_m . We call this the One-more Bit Problem.

Thus, we consider the following scenario. First, $\alpha = (\alpha_1, \ldots, \alpha_m)$ is transmitted. The Eavesdropper may compute any function $\eta = A_1(\alpha)$, such that $|\eta| = n/5$, and retain η alone. Next, she is provided with s and X_i for all $i = 1, \ldots, m-1$. She must now guess X_m .

As above, let A_2 be the algorithm the Eavesdropper uses to guess X_m , given η , s and the X_i 's. Denote $X^- = (X_1, \ldots, X_{m-1})$ and $\alpha^- = (\alpha^{(1)}, \ldots, \alpha^{(m-1)})$. Note that X^- is fully determined by α^- and s. Thus, we write $X^- = X^-(\alpha^-, s)$.

Using A_2 we can construct another algorithm \hat{A}_2 which, given $\eta = A_1(\alpha)$, s and α^- guesses X_m . (The difference between A_2 and \hat{A}_2 is that \hat{A}_2 gets α^- as input, instead of X^- .) Algorithm \hat{A}_2 works as follows. First, \hat{A}_2 computes $X^- = X^-(s, \alpha^-)$. Then, it runs $A_2(\eta, s, X^-)$. Thus, the success probability of any algorithm \hat{A}_2 is at most the success of the best algorithm \hat{A}_2 . We now bound the success probability of \hat{A}_2 .

For a given η and α^- , let $B = B_{\eta,\alpha^-} = (\hat{A}_2(\eta, s_1, \alpha^-), \dots, \hat{A}_2(\eta, s_K, \alpha^-))$. In other words, B is the enumeration of answers of \hat{A}_2 , for the given η and α^- . Set

$$L_B = \left\{ \alpha_m : |\bar{B} \cdot v(\alpha_m)| \ge \frac{2K}{2^{0.4k/2}} \right\}.$$

By Lemma 5, $|L_B| \leq 2^{cn+0.4k}$. Thus,

Lemma 7. If $\alpha \in L_{B_{\eta,\alpha^-}}$, for some η , then $C(\alpha|\hat{A}_2) \leq (m-1)n + \frac{n}{5} + cn + 0.4k$.

Proof. The sequence α is composed of α^- together with $\alpha^{(m)}$. Thus, given \hat{A}_2 the entire sequence α can be fully characterized by:

- 1. η n/5 bits,
- 2. α^- (m-1)n bits, and
- 3. The index of α_m in $L_{B_{n,\alpha}}$ cn + 0.4k bits,

The construction of α from this information is analogous to that given in the proof of Corollary 6.

Corollary 8 For any A_1 , A_2 ,

$$\Pr\left[A_2(A_1(\alpha), s, X^-) = X_m\right] \le \frac{1}{2} + \frac{2}{2^{k/5}}.$$

Proof. By Lemma 7 and the definition of L_B , if $C(\alpha|\hat{A}_2) > (m-1)n + \frac{n}{5} + cn + 0.4k$, then, for any η

$$\Pr\left[\hat{A}_2(\eta, s, \alpha^-) = X_m)\right] < \frac{1}{2} + \frac{1}{2^{0.4k/2}}.$$

Thus, we also have that for such α and any algorithm A_2 (which gets X^- instead of α^- as input) and $\eta = A_1(\alpha)$,

$$\Pr\left[A_2(A_1(\alpha), s, X^-) = X_m\right] < \frac{1}{2} + \frac{1}{2^{0.4k/2}}.$$

For $\alpha \in \{0,1\}^{nm}$,

$$\Pr\left[C(\alpha|\hat{A}_2) \le (m-1)n + \frac{n}{5} + cn + 0.4k\right] \ll \frac{1}{2^{0.4k/2}}.$$

Thus, in all,

$$\Pr\left[A_2(A_1(\alpha), s, X^-) = X_m\right] \le \frac{1}{2} + \frac{2}{2^{0.2k}}.$$

3.4 Multi-bit Security - Any Two Messages

We consider the case of distinguishing two messages. Let $M^{(0)}$, $M^{(1)}$ be two distinct messages, $M^{(i)} \in \{0,1\}^m$. We show that if an eavesdropper's algorithm A can distinguish between $M^{(0)}$ and $M^{(1)}$, with probability p, then there is another algorithm $B = (B_1, B_2)$, which solves the One-more Bit Problem with probability $\geq 1/2 + p/2$. Specifically, w.l.o.g. assume that

$$\Pr\left[A(M^{(1)}) = 1\right] - \Pr\left[A(M^{(0)}) = 1\right] = p \ge 0.$$

We construct $B(B_1, B_2)$, such that for $\beta \in \{0, 1\}^{mn}$,

$$\Pr\left[B_2(B_1(\beta), s, X^{-}(\beta, s)) = s(\beta^{(m)})\right] \ge \frac{1}{2} + \frac{p}{2}.$$

Suppose that $M^{(i)} \neq 0$, for i = 0, 1. Let P be an $m \times m$ non-singular matrix over F_2 such that $P \cdot M^{(0)} = e_1$ and $P \cdot M^{(1)} = e_1 + e_m$, where e_1 and e_m are the unit vectors with a 1 in the first and last coordinates, respectively. For $\beta = (\beta^{(1)}, \ldots, \beta^{(m)})$ we view β as an $m \times n$ matrix. Thus, $P \cdot \beta$ is another $m \times n$ matrix. A detailed description of $B = (B_1, B_2)$, given $A = (A_1, A_2)$, is provided hereunder. For the case that $M^{(0)} = 0$, an analogous proof works, omitting e_1 in all its appearances.

Input: β . Output: η .

 $\alpha := P^{-1}\beta.$

 2 $\eta := A_1(\alpha).$

Input: η , s and $X^- = X^-(s, \beta)$. Output: $X_m = s(\beta^{(m)})$. $\frac{B_2:}{1}$

Choose $r \in \{0, 1\}$ at random.

2 $X := X^- \circ 0$ (concatenation).

3 $Z = X \oplus e_1 \oplus re_m$.

 $Y := P^{-1}Z.$ 4

Output $A_2(\eta, s, Y) \oplus r$.

First we prove a technical lemma.

Lemma 9. Suppose that

$$\Pr\left[A(M^{(1)}) = 1\right] - \Pr\left[A(M^{(0)}) = 1\right] = p \ge 0.$$

Consider choosing $r \in \{0,1\}$ at random and then running A on $M^{(r)}$. Then

$$\Pr\left[A(M^{(r)}) = r\right] = \frac{1}{2} + \frac{p}{2}.$$

Proof.

$$\begin{split} \Pr\left[A(M^{(r)}) = r\right] = \\ \Pr\left[A(M^{(1)}) = 1\right] \Pr\left[r = 1\right] + \Pr\left[A(M^{(0)}) = 0\right] \Pr\left[r = 0\right] = \\ \frac{1}{2} \left(\Pr\left[A(M^{(1)}) = 1\right] + \left(1 - \Pr\left[A(M^{(0)}) = 1\right]\right)\right) = \frac{1}{2} + \frac{p}{2}. \end{split}$$

Lemma 10.

$$\Pr\left[B_2(B_1(\beta), s, X^{-}(\beta, s)) = s(\beta^{(m)})\right] = \frac{1}{2} + \frac{p}{2}.$$

Proof. Set $\delta = s(\beta^{(m)})$, and $\beta^- = (\beta^1, \dots, \beta^{m-1})$. By construction $Z = (s(\beta^-))$ $(\circ 0) \oplus (e_1 \oplus re_m) = (s(\beta) \oplus \delta e_m) \oplus (e_1 \oplus re_m) = s(\beta) \oplus (e_1 \oplus (r \oplus \delta) e_m)$. Also, by construction, $\alpha = P^{-1}\beta$ (B_1 line 1), and $P^{-1}e_1 = M^{(0)}$ and $P^{-1}(e_1 \oplus e_m) = M^{(1)}$. Thus, $P^{-1}(e_1 \oplus ((r \oplus \delta)e_m)) = M^{(r \oplus \delta)}$. Thus,

$$Y = P^{-1}Z = P^{-1}(s(\beta) \oplus (e_1 \oplus (r \oplus \delta)e_m)) = s(P^{-1}\beta) \oplus P^{-1}(e_1 \oplus (r \oplus \delta)e_m)$$
$$= s(\alpha) \oplus M^{(r \oplus \delta)}.$$

By construction, $B_1(\beta) = A_1(\alpha)$, and $B_2(\eta, s, X^-(s, \beta)) = A_2(\eta, s, Y) \oplus r$. Thus,

$$\Pr\left[B_2(B_1(\beta), s, X^-(s, \beta)) = \delta\right] = \Pr\left[A_2(A_1(\alpha), s, s(\alpha) \oplus M^{(r \oplus \delta)}) = r \oplus \delta\right] = \Pr\left[A(M^{(r \oplus \delta)}) = r \oplus \delta\right] \le \frac{1}{2} + \frac{p}{2}.$$

Together with Corollary 8 we get

Corollary 11 For any $M^{(0)}$ and $M^{(1)}$,

$$\left| \Pr \left[A(M^{(1)}) = 1 \right] - \Pr \left[A(M^{(0)}) = 1 \right] \right| \le \frac{1}{2^{k/5}}.$$

By Lemma 1 this completes the proof of Theorem 1.

References

- Y. Aumann and U. Feige. One message proof systems with known space verifies. In D.P. Stinson, editor, Advances in Cryptology, pages 85–99. Springer-Verlag, 1993.
- 2. C. H. Bennett, G. Brassard, C. Crepeau, and U. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6), 1995.
- C. Cachin. Entropy Measures and Unconditional Security in Cryptography, volume 1. Hartung-Gorre Verlag, Konstaz, Germany, 1997.
- C. Cachin and U. M. Maurer. Unconditional security against memory bounded adversaries. In Proceedings of Crypto '97, 1997.
- A. De-Santis, G. Persiano, and M. Yung. One-message statistical zero-knowledge proofs with space-bounded verifier. In *Proceedings of the 19th ICALP*, 1992.
- M. Li and P. M. B. Vitanyi. An Introduction to Kolmogorov Complexity and Its Applications. Springer-Verlag, New York, 2nd edition, 1997.
- U. M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5:53–66, 1992.
- 8. U. M. Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, 39:733–742, 1993.
- R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, 1995.
- C. E. Shannon. Communication theory of secrecy systems. Bell Systems Technical Journal, 28:656–715, 1949.

The All-or-Nothing Nature of Two-Party Secure Computation

Amos Beimel¹, Tal Malkin^{2,3}, and Silvio Micali²

Division of Engineering and Applied Sciences Harvard University, 40 Oxford st., Cambridge, MA 02138 beimel@deas.harvard.edu

Supported by grants ONR-N00014-96-1-0550 and ARO-DAAL03-92-G0115.

Laboratory for Computer Science, Massachusetts Institute of Technology 545 Technology sq., Cambridge, MA 02139

atal@theory.lcs.mit.edu

Supported by DARPA grant DABT63-96-C-0018.

Abstract. A function f is computationally securely computable if two computationally-bounded parties Alice, having a secret input x, and Bob, having a secret input y, can talk back and forth so that (even if one of them is malicious) (1) Bob learns essentially only f(x,y) while (2) Alice learns essentially nothing.

We prove that, if any non-trivial function can be so computed, then so can every function. Consequently, the complexity assumptions sufficient and/or required for computationally securely computing f are the same for every non-trivial function f.

1 Introduction

SECURE COMPUTATION. Let f be a two-argument finite function, that is, $f: S_1 \times S_2 \to S_3$ (where S_1, S_2 , and S_3 are finite sets), and let Alice and Bob be two possibly malicious parties, the first having a secret input $x \in S_1$ and the second having a secret input $y \in S_2$. Intuitively, securely computing f means that Alice and Bob keep turns exchanging message strings so that (1) Bob learns the value z = f(x, y), but nothing about x (which is not already implied by z and y), no matter how he cheats, while (2) Alice learns nothing about y (and thus nothing about z not already implied by x), no matter how she cheats.

In a sense, therefore, a secure computation of f has two constraints: a correctness constraint, requiring that Bob learns the correct value of f(x, y), and a privacy constraint, requiring that neither party learns more than he/she should about the other's input.

Throughout this paper, any function to be securely computed is a finite, two-argument function.

THE ONE-SIDEDNESS OF SECURE COMPUTATION. The notion of secure computation informally recalled above is the traditional one used in the two-party, *malicious* model (cf., [GMW87, Section 4.2], and [Kil88,Kil90]). This notion is

"one-sided" in that only Bob learns the result of computing f, while Alice learns nothing. Such one-sidedness is unavoidable in our malicious model. In principle, one could conceive of a more general notion of secure computation in which "both Alice and Bob learn $f(x,y)^1$." However, such a more general notion is not achievable in a two-party, malicious model: the first party who gets the desired result, if malicious, may stop executing the prescribed protocol, thus preventing the other from learning f(x,y). Moreover, such a malicious party can terminate prematurely the execution of the prescribed protocol exactly when he/she "does not like" the result.

Trivial and Non-Trivial Functions. A function f is called trivial if it can be securely computed even if a cheating party has unlimited computational power, and non-trivial otherwise.

An example of a trivial function is the "projection of the first input"; namely the function $P_1: \{0,1\} \times \{0,1\} \to \{0,1\}$ so defined: $P_1(b_0,b_1) = b_0$. Another example is the "exclusive-or function"; namely, the function XOR: $\{0,1\} \times \{0,1\} \to \{0,1\}$ so defined: $XOR(b_0,b_1) = b_0 + b_1 \mod 2$. Indeed, a secure way of computing either function consists of having Alice send her secret bit to Bob. This elementary protocol clearly is a correct and private way of computing P_1 . It also is a correct and private way of computing XOR. Indeed, Alice's revealing her own secret bit b_0 enables Bob to compute locally and correctly the desired XOR of b_0 and b_1 . Moreover, Alice's revealing b_0 also satisfies the privacy constraint: Bob could deduce Alice's bit anyway from the output of the XOR function he is required to learn.

An example of a non-trivial function is the function AND: $\{0,1\} \times \{0,1\} \rightarrow \{0,1\}$ so defined: AND $(b_0,b_1)=b_0 \wedge b_1$. Another non-trivial function is the (chosen 1-out-of-2) oblivious transfer; namely, the function OT: $\{0,1\}^2 \times \{0,1\} \rightarrow \{0,1\}$ so defined: OT $((b_0,b_1),i)=b_i$, that is, Bob only learns the bit of Alice he chooses. (The non-triviality of these functions follows from [CK89].)

Secure Computability of Non-Trivial Functions. By definition, securely computing non-trivial functions is *conceivable* only when (at least one of) Alice and Bob are computationally bounded, but by no means *guaranteed*. Nonetheless, a series of results have established that secure computation of non-trivial functions is possible under complexity assumptions of various strengths. In particular,

 The OT function is securely computable under the assumption that integer factorization is computationally hard [Rab81,FMR84,EGL85,Cré88].³

 $^{^{1}}$ Or even a more general scenario where Bob learns f(x,y) while Alice learns g(x,y).

² Or g(x,y) in the more general scenario.

³ Rabin [Rab81] introduced a variant of the oblivious transfer, the *random oblivious transfer*, and provided an implementation of it which is provably secure in the honest-but-curious model. Fischer, Micali, and Rackoff [FMR84] improved his protocol so as to be provably secure against malicious parties. Even, Goldreich, and

- All functions are securely computable if factoring is hard [Yao86]; and, actually,
- All functions are securely computable if $\it any$ trapdoor permutation exists [GMW87]. 4

Such results raise fundamental questions about the strength of the computational assumptions required for secure computation. In particular,

- Q1: What assumption is required for securely computing at least one non-trivial function?
- Q2: What assumption is required for securely computing a given non-trivial function f?
- Q3: Are there assumptions sufficient for securely computing some non-trivial function f but not sufficient for securely computing some other non-trivial function g?

COMPLETENESS FOR SECURE COMPUTATION. Another important result is that the OT function is *complete* for secure computation [Kil88]⁵. By this we mean that, if OT is securely computable, then so are all functions. A bit more specifically, given any function f and any protocol securely computing OT, one can efficiently and uniformly construct a protocol securely computing f.

The completeness of the OT function raises additional fundamental questions. In particular,

Q4: Are there other (natural) functions that are complete for secure computation?
Q5: Is there a (natural) characterization of the functions complete for secure computation?

1.1 Main Results

A CHARACTERIZATION OF COMPLETE FUNCTIONS. In this paper we prove the following

Main Theorem: Any non-trivial function is complete for secure computation.

Lempel [EGL85] introduced the notion of the chosen 1-out-of-2 oblivious transfer, together with an implementation of it which is provably secure in the honest-but-curious model. Finally, Crépeau [Cré88] showed how to transform any secure protocol for the random oblivious transfer to a secure protocol for the chosen 1-out-of-2 oblivious transfer.

⁴ The hardness of factoring implies the existence of trapdoor permutations, but the vice-versa might not hold.

⁵ Kilian [Kil91] also proves a more general result, but in a different model, which we discuss in Subsection 1.2.

Clearly, our result provides an explicit and positive answer to questions Q4 and Q5, and an explicit and negative answer to Q3. Our result also provides an *implicit* answer to questions Q1 and Q2. Namely, letting f be any given non-trivial function, and A_f be the assumption that f is securely computable:

For any non-trivial function g, assumption A_f is both necessary and sufficient for securely computing g.

AN INTERPRETATION OF OUR MAIN THEOREM. Our main theorem also suggests that just assuming the existence of one-way functions may be *insufficient* to guarantee secure computation. Let us explain. Impagliazzo and Rudich [IR89] show that, without also proving that $\mathcal{P} \neq \mathcal{NP}$, no protocol having oracle-access to a random function can be proved to compute the OT function securely. This result has been interpreted as providing strong evidence that "one-way functions are not sufficient for constructing a protocol securely computing the OT function." It is then according to the same interpretation that our main theorem suggests that, for *any* non-trivial function f, A_f should be stronger than the existence of one-way functions.

A CHARACTERIZATION OF TRIVIAL FUNCTIONS. Is there a combinatorial property that makes a two-argument function securely computable by two, possibly malicious, parties with unbounded computing power? In our paper we also provide such a characterization (actually crucial to the proof of our main theorem⁶) in terms of *insecure minors*.

We say that f contains an insecure minor if there exist inputs x_0, y_0, x_1, y_1 such that $f(x_0, y_0) = f(x_1, y_0)$ and $f(x_0, y_1) \neq f(x_1, y_1)$, and prove:

Main Lemma: A two-argument function f is trivial if and only if f does not contain an insecure minor.

1.2 Comparison to Previous Work

THE HONEST-BUT-CURIOUS MODEL. Both completeness and characterization of non-trivial functions have been extensively investigated with respect to a weaker notion of two-party secure computation introduced in [GMW87]: the honest-but-curious model⁷. In this model, the parties are guaranteed to properly execute a prescribed protocol, but, at the end of it, each of them can use his/her own view of the execution to infer all he/she can about the other's input. In this model, because no protocol can be prematurely terminated, it is meaningful to consider "two-sided" secure computation of a function f; that is, one in which each party learns f(x,y), but nothing else about the other's input that is not

⁶ Note that our main theorem provides a characterization of both trivial and non-trivial functions, though not a combinatorial-looking one!

⁷ Originally called "the semi-honest model" in [GMW87].

already implicit in f(x, y) and his/her own input. Indeed this is the traditional notion of secure function computation in the honest-but-curious model.

Similar to the malicious model, a function is said to be trivial in the honest-but-curious model if it can be securely computed even if the two (honest-but-curious) parties have unbounded computing power, and non-trivial otherwise. The above mentioned results of [Yao86,GMW87] immediately imply that every two-argument function is securely computable in the honest-but-curious model, under the corresponding complexity assumptions (hardness of factoring and existence of trapdoor permutations).

A combinatorial characterization of the trivial functions in the honest-butcurious model was first given by Chor and Kushilevitz [CK89] for *Boolean* functions (i.e., predicates), and then by Kushilevitz [Kus89] for all functions.

While in the malicious model we prove that all non-trivial functions are complete, in the honest-but-curious one the "corresponding" theorem does not hold; there exists a (non-Boolean) function that is neither trivial nor complete [Kus89,Kil91,KKMO98].⁸ On the other hand, Kilian, Kushilevitz, Micali, and Ostrovsky [KKMO98] prove that any non-trivial *Boolean* function is complete in the honest-but-curious model.

KILIAN'S MODEL. In [Kil91] Kilian characterizes the functions f that are complete in an hybrid model of secure computation. Namely, the functions f for which, given access to a two-sided black-box for f (i.e., one giving the result f(x,y) to both Alice and Bob), one can construct, for any function, a one-sided protocol that is information-theoretically secure against unbounded malicious parties. He proves that these functions f are exactly those containing an embedded-or, a special case of our insecure minor (i.e., one satisfying the additional constraint $f(x_0, y_0) = f(x_0, y_1)$).

In sum, Kilian's result "reduces" standard (one-sided) protocols to two-sided black boxes. Notice that this is different (and not applicable) to our case, where we reduce standard protocols to standard protocols. (Indeed, our characterization of the complete function is different, and there are functions that are complete in our setting but not in his.)

Also notice that two-sided black boxes might be implementable via "tamper-proof hardware" or in some other physical model, but, as explained above, no protocol can securely implement a two-sided black box for a function f against malicious parties.⁹

⁸ [KKMO98] prove this by combining the following two results. [Kus89] shows an example of a function which is non-trivial yet does not contain an embedded or, and [Kil91] shows that a function that does not contain an embedded or cannot be complete in this model. We note that this example is a function which contains an insecure minor, and thus *is* complete in the malicious (one-sided) model, as we prove in this paper.

⁹ Two-sided boxes may instead be implemented by protocols (under certain complexity assumptions) in the honest-but-curious model.

REDUCTION MODELS. Black-box reductions (as those of [CK88,Kil91,KKMO98]) are an elegant way to build new secure protocols. While two-sided boxes are not implementable by secure protocols against malicious parties, one-sided black boxes can be (under certain complexity assumptions). Thus, one may consider completeness under one-sided black box reductions. However, as we shall point out in Section 4.2, such reductions are not strong enough to solve the questions we are interested in. We thus use an alternative definition of a reduction that is natural for protocols secure against bounded malicious parties. Informally, for us a reduction is a transformation of a given secure protocol for f (rather than a one-sided black box for f) into a protocol for g secure against computationally bounded malicious parties.

Organization. In Section 2 we define protocols, and secure computation in the unbounded honest-but-curious model. In Section 3 we provide a definition of secure computation in the unbounded malicious model, and proceed to characterize the trivial functions. Finally, in Section 4 we characterize the complete functions, and prove that any non-trivial function is complete.

2 Preliminaries

2.1 Protocols

Following [GMR85], we consider a two-party protocol as a pair, (A, B), of Interactive Turing Machines (ITMs for short). Briefly, on input (x, y), where x is a private input for A and y a private input for B, and $random input (r_A, r_B)$, where r_A is a private random tape for A and r_B a private random tape for B, protocol (A, B) computes in a sequence of rounds, alternating between A-rounds and B-rounds. In an A-round (B-round) only A (only B) is active and sends a message (i.e., a string) that will become an available input to B (to A) in the next B-round (A-round). A computation of (A, B) ends in a B-round in which B sends the empty message and computes a private output.

TRANSCRIPTS, VIEWS, AND OUTPUTS. Letting E be an execution of protocol (A, B) on input (x, y) and random input (r_A, r_B) , we define:

- The transcript of E consists of the sequence of messages exchanged by A and B, and denoted by TRANS^{A,B} (x, r_A, y, r_B)
- The view of A consists of the triplet (x, r_A, t) , where t is E's transcript, and denoted by VIEW_A^{A,B} (x, r_A, y, r_B) ;
- The view of B consists of the triplet (y, r_B, t) , where t is E's transcript, and denoted by VIEW_B^{A,B} (x, r_A, y, r_B) ;
- The output of E consists of the string z output by B in the last round of E, and denoted by $\mathrm{OUT}_B(y,r_B,t)$, where t is E's transcript.

 $^{^{10}\,}$ Due to the one-sidedness of secure computation, only machine B produces an output.

In all the above the superscript (A, B) will sometimes be omitted when clear from the context.

We consider the random variables $\operatorname{TRANS}(x,\cdot,y,r_B)$, $\operatorname{TRANS}(x,r_A,y,\cdot)$ and $\operatorname{TRANS}(x,\cdot,y,\cdot)$, respectively obtained by randomly selecting r_A , r_B , or both, and then outputting $\operatorname{TRANS}(x,r_A,y,r_B)$. We also consider the similarly defined random variables $\operatorname{VIEW}_A(x,\cdot,y,r_B)$, $\operatorname{VIEW}_A(x,r_A,y,\cdot)$, $\operatorname{VIEW}_B(x,\cdot,y,r_B)$, $\operatorname{VIEW}_B(x,r_A,y,\cdot)$, and $\operatorname{VIEW}_B(x,\cdot,y,\cdot)$,

2.2 Secure Computation in the Unbounded Honest-but-Curious Model

Among all notions of secure computation, the one for two unbounded honestbut-curious parties is the simplest one to formalize. In this model the parties Alice and Bob are guaranteed to follow the prescribed protocol (A, B) (namely they use the right ITMs A and B), but may try to obtain as much information as they can from their own views. Intuitively, a protocol is secure in this model if the following conditions hold: (1) Bob learns the value z = f(x, y), but nothing about x (not already implied by z and y), while (2) Alice learns nothing about y(and thus nothing about z not already implied by x). A formal definition follows.

Definition 1. Let $f: S_1 \times S_2 \to S_3$ be a finite function. A protocol (A, B) securely computes f against unbounded honest-but-curious parties, if the following conditions hold:

1. Correctness: $\forall x \in S_1, \forall y \in S_2, \forall r_A, \forall r_B, letting v = \text{VIEW}_B^{A,B}(x, r_A, y, r_B),$

$$OUT_B(v) = f(x, y).$$

2. Privacy:

Alice's Privacy:
$$\forall x_0, x_1 \in S_1, \forall y \in S_2, \forall r_B, if f(x_0, y) = f(x_1, y) then$$

$$VIEW_B^{A,B}(x_0, \cdot, y, r_B) = VIEW_B^{A,B}(x_1, \cdot, y, r_B).^{11}$$

Bob's Privacy:
$$\forall x \in S_1, \forall y_0, y_1 \in S_2, \forall r_A,$$

 $\text{VIEW}_A^{A,B}(x, r_A, y_0, \cdot) = \text{VIEW}_A^{A,B}(x, r_A, y_1, \cdot).$

3 A Combinatorial Characterization of Trivial Functions

So far, we have intuitively defined a trivial function to be one that is computable by a protocol that is *secure against unbounded malicious parties*. ¹² Combinatorially characterizing trivial functions, however, requires first a quite formal notion of secure computation in our setting, a task not previously tackled. This is what we do below.

¹¹ Equivalently, the corresponding transcripts are identically distributed (and similarly below).

By this we do not mean that the parties participating in a protocol computing a trivial function are computationally-unbounded, but that the "privacy and correctness" of their computation holds even when one of them is allowed to be malicious and computationally-unbounded.

3.1Secure Computation in the Unbounded Malicious Model

In this model Alice or Bob may be malicious, namely cheat in an arbitrary way, not using the intended ITM A (or B), but rather an arbitrary (computationally unbounded) strategy A' (or B') of their choice. The definition of secure computation in the malicious model requires some care. For example, it is not clear how to define what the input of a malicious party is.

We handle the definition of secure computation in the spirit of [MR92] (a definition primarily aimed at secure computation in a multi-party scenario, such as [BGW88,CCD88]). Intuitively, we require that when Alice and Bob are honest then Bob computes the function f correctly relative to his own input and Alice's input. We also require that when Bob is honest and for any possible malicious behavior of Alice, Bob computes the function f correctly relative to his own input and Alice's input as defined by evaluating a predetermined input function on Alice's view of the joint computation. Because the computation is one-sided and a malicious Bob might not output any value, the correctness requirement is limited to the above two cases. Finally, we require privacy for an honest Alice against a possibly malicious Bob, and privacy for an honest Bob against a possibly malicious Alice.

Definition 2. Let $f: S_1 \times S_2 \to S_3$ be a finite function. A protocol (A, B)securely computes f against unbounded malicious parties, if the following conditions hold:

1. Correctness: $\forall x \in S_1, \forall y \in S_2, \forall r_A, \forall r_B,$

Correctness when both Alice and Bob are honest: $Letting\ v = VIEW_B^{A,B}(x,r_A,y,r_B),\ then\ OUT_B(v) = f(x,y).$

Correctness when only Bob is honest: For every strategy A' there is $\mathcal{I}_{A'}: \{0,1\}^* \to S_1 \text{ such that, letting}$

$$v_{A'}' = \text{VIEW}_{A'}^{A',B}(x,r_A,y,r_B) \text{ and } v_B' = \text{VIEW}_{B}^{A',B}(x,r_A,y,r_B),$$

$$OUT_B(v'_B) = f(\mathcal{I}_{A'}(v'_{A'}), y).^{13}$$

2. Privacy:

Alice's Privacy: For every strategy B', $\forall x_0, x_1 \in S_1, \forall y \in S_2, \forall r_B$, if

$$f(x_0, y) = f(x_1, y)$$

By the previous condition, the mapping \mathcal{I}_A (i.e., for honest Alice) gives a "correct" input, which is either x itself or "equivalent" to x, in the sense that it yields the same output f(x,y). Notice that for secure computation of a function f we may restrict the protocol so that Bob always outus a value that is compatable with his input. That is, on input y Bob outputs a value z such that there is some x for which f(x,y)=z(indeed, Bob before outputing z can always check for compatibility and output f(0,y) otherwise). When restricted to these secure computation the correctness for honest Bob and Alice implies the correctness when only Bob is honest, that is, the function $\mathcal{I}_{A'}$ is guaranteed to exist.

then

$$VIEW_{B'}^{A,B'}(x_0, \cdot, y, r_B) = VIEW_{B'}^{A,B'}(x_1, \cdot, y, r_B).$$

Bob's Privacy: For every strategy A', $\forall x \in S_1, \forall y_0, y_1 \in S_2, \forall r_A$,

$$VIEW_{A'}^{A',B}(x, r_A, y_0, \cdot) = VIEW_{A'}^{A',B}(x, r_A, y_1, \cdot).$$

Note that security against malicious parties implies security against honestbut-curious parties. That is,

Fact 1 If a protocol securely computes the function f in the unbounded malicious model, it securely computes f in the unbounded honest-but-curious model.

Definition 3 (Trivial and non-trivial functions). A finite function f is called trivial if there exists a protocol securely computing it in the unbounded malicious model; otherwise, f is called non-trivial.

3.2 The Combinatorial Characterization

We prove that the trivial functions are exactly those that do not contain an insecure minor (a simple generalization of an *embedded or* $[CK89]^{14}$).

Definition 4 (Insecure minor). A function $f: S_1 \times S_2 \to S_3$ contains an insecure minor if there exist $x_0, x_1 \in S_1$, $y_0, y_1 \in S_2$, and $a, b, c \in S_3$ such that $b \neq c$, and $f(x_0, y_0) = f(x_1, y_0) = a$, $f(x_0, y_1) = b$, and $f(x_1, y_1) = c$. Graphically, ¹⁵

$$f: \mathbf{x_0} | \mathbf{x_1} \\
 \mathbf{y_0} | a | a \\
 \mathbf{y_1} | b | c$$

Examples. As immediately apparent from their tables, each of the AND and OT functions contain an insecure minor (and actually an embedded or):

AND:	0	1
0	0	0
1	0	1

OT :	(0, 0)	(0, 1)
0	0	0
1	0	1

Theorem 1. A function $f(\cdot, \cdot)$ is trivial if and only if f does not contain an insecure minor.

An embedded or is an insecure minor in which a = b. As shown in [CK89], having an embedded or implies non-triviality in the two-sided *honest-but-curious* model, and characterizes the *Boolean* non-trivial functions in this model.

¹⁵ This graphical convention will be used in the rest of the paper, namely a table where columns correspond to possible inputs for Alice, rows correspond to possible inputs for Bob, and the entries are the corresponding output values.

Proof. We break the proof into two parts; Theorem 1 follows from the following Claim 1 and Claim 2.

First, we assume that f does not contain an insecure minor and prove that f is trivial by constructing a protocol (A, B) that securely computes f against malicious unbounded parties. Fix any $x_0 \in S_1$ and $y_0 \in S_2$. The protocol (A, B), described in Fig. 1, has a single round of communication (one message sent from A to B), and is deterministic (namely A and B ignore their random inputs).

Claim 1. If f does not contain an insecure minor then it is trivial.

Proof. We prove our claim by showing that f is securely computed against unbounded malicious parties by the following protocol (A, B) described in Fig. 1.

```
Protocol (A, B)

A, on input x \in S_1:

send to Bob the message a \stackrel{\text{def}}{=} f(x, y_0).

B, on input y \in S_2, upon receipt of the message a from Alice:

Find the lexicographically first x_1 such that f(x_1, y_0) = a

Set the output \text{OUT}_B(y, r_B, a) to f(x_1, y).

If no such x_1 exists, set the output \text{OUT}_B(y, r_B, a) to f(x_0, y).
```

Fig. 1. A secure protocol (against unbounded malicious parties) for a function f not containing an insecure minor.

We first prove the correctness of the protocol. Recall that x and y are the inputs held by honest Alice and honest Bob respectively. Correctness when both parties are honest follows since for any message $a = f(x, y_0)$ sent by honest Alice, an honest Bob finds x_1 such that $f(x, y_0) = f(x_1, y_0)$. Since f does not contain an insecure minor then it must hold that $f(x, y) = f(x_1, y)$ (otherwise x, x_1, y_0, y constitute an insecure minor). Thus, Bob's output $-f(x_1, y)$ – is correct.

To prove correctness when only Bob is honest, we first define the following input function $\mathcal{I}_{A'}: \{0,1\}^* \to S_3$ where if there is no x_1 such that $a = f(x_1, y_0)$ then $\mathcal{I}_{A'}(x, r_A, a) = x_0$ and otherwise $\mathcal{I}_{A'}(x, r_A, a)$ is the lexicographically first x_1 such that $a = f(x_1, y_0)$. Notice that the input function $\mathcal{I}_{A'}$ is the same for every adversary A'. By the definition of $\mathcal{I}_{A'}$ it always holds that $\text{OUT}_B(y, r_B, a) = f(\mathcal{I}_{A'}(x, r_A, a), y)$ and correctness follows.

Alice's privacy follows by observing that all information sent to Bob, namely $f(x, y_0)$, can be computed by Bob alone from the output of the function f(x, y). This is because Bob can find some x' such that f(x', y) = f(x, y), and conclude that $f(x, y_0) = f(x', y_0)$ (since f does not contain an insecure minor). Bob's privacy follows immediately from the fact that this is a one-round protocol, where Bob sends no information to Alice, and thus her view is clearly identical for any input he may have.

Let us now prove the second part of Theorem 1.

Claim 2. If a function f is trivial then it does not contain an insecure minor.

Proof. If f is trivial, then there is a protocol (A, B) securely computing f against unbounded parties. In particular, by Fact 1, Protocol (A, B) securely computes f against honest-but-curious unbounded parties. We assume, for sake of contradiction, that f contains an insecure minor.

Let x_0, x_1, y_0, y_1 constitute an insecure minor of f, that is there are $a, b, c \in S_3$ such that $b \neq c$ and

$$f: \mathbf{x_0} | \mathbf{x_1} \\
 \mathbf{y_0} | a | a \\
 \mathbf{y_1} | b | c$$

By Bob's privacy from Definition 1,

$$VIEW_A(x_0, r_A, y_1, \cdot) = VIEW_A(x_0, r_A, y_0, \cdot)$$

for every r_A . By ranging over all possible r_A we get

$$TRANS(x_0, \cdot, y_1, \cdot) = TRANS(x_0, \cdot, y_0, \cdot).$$

On the other hand, by Alice's privacy, since $f(x_0, y_0) = f(x_1, y_0) = a$,

$$VIEW_B(x_0, \cdot, y_0, r_B) = VIEW_B(x_1, \cdot, y_0, r_B)$$

for every r_B . Again, by ranging over all possible r_B we get

$$TRANS(x_0, \cdot, y_0, \cdot) = TRANS(x_1, \cdot, y_0, \cdot).$$

Finally, again by Bob's privacy

$$TRANS(x_1, \cdot, y_0, \cdot) = TRANS(x_1, \cdot, y_1, \cdot).$$

Thus, by transitivity,

$$TRANS(x_0, \cdot, y_1, \cdot) = TRANS(x_1, \cdot, y_1, \cdot). \tag{1}$$

We next use the following proposition, proved by [CK89] to hold for every protocol, to argue that this transcript is equally distributed even if we fix the random input of Bob.

Proposition 1. Let $u_0, u_1, v_0, v_1, r_{A,0}, r_{A,1}, r_{B,0}, r_{B,1}$ be inputs and random inputs such that

$$TRANS(u_0, r_{A,0}, v_0, r_{B,0}) = TRANS(u_1, r_{A,1}, v_1, r_{B,1}) = t.$$

Then,
$$TRANS(u_0, r_{A,0}, v_1, r_{B,1}) = TRANS(u_1, r_{A,1}, v_0, r_{B,0}) = t.$$

In other words, if changing the inputs for both Alice and Bob yields the same transcript, then changing the input for Alice only (or Bob only) will also yield the same transcript.

Fix arbitrary random inputs q_A and q_B , and let $t = \text{TRANS}(x_0, q_A, y_1, q_B)$. By Equation (1), there exist q'_A, q'_B such that $\text{TRANS}(x_1, q'_A, y_1, q'_B) = t$, which by Proposition 1 implies that $t = \text{TRANS}(x_1, q'_A, y_1, q_B)$. Now, by Proposition 1, it holds that

$$\Pr_{r_A, r_B} \left[\text{TRANS}(x_0, \cdot, y_1, \cdot) = t \right] \\
= \Pr_{r_A} \left[\text{TRANS}(x_0, \cdot, y_1, q_B) = t \right] \cdot \Pr_{r_B} \left[\text{TRANS}(x_0, q_A, y_1, \cdot) = t \right], \quad (2)$$

and similarly

$$\Pr_{r_A, r_B} \left[\text{TRANS}(x_1, \cdot, y_1, \cdot) = t \right] \\
= \Pr_{r_A} \left[\text{TRANS}(x_1, \cdot, y_1, q_B) = t \right] \cdot \Pr_{r_B} \left[\text{TRANS}(x_1, q'_A, y_1, \cdot) = t \right].$$
(3)

By Proposition 1, for every r_B

$$TRANS(x_0, q_A, y_1, r_B) = t$$
 if and only if $TRANS(x_1, q'_A, y_1, r_B) = t$

Hence,

$$\Pr_{r_B} [\text{TRANS}(x_0, q_A, y_1, \cdot) = t] = \Pr_{r_B} [\text{TRANS}(x_1, q'_A, y_1, \cdot) = t].$$
 (4)

Since by (1) $\Pr_{r_A,r_B} \left[\operatorname{TRANS}(x_0,\cdot,y_1,\cdot) = t \right] = \Pr_{r_A,r_B} \left[\operatorname{TRANS}(x_1,\cdot,y_1,\cdot) = t \right]$, and from (2),(3),(4), we get that, for every q_B and t,

$$\Pr_{r_A}\left[\mathrm{TRANS}(x_0,\cdot,y_1,q_B)=t\right] \ = \ \Pr_{r_A}\left[\mathrm{TRANS}(x_1,\cdot,y_1,q_B)=t\right].$$

That is,

$$TRANS(x_0, \cdot, y_1, q_B) = TRANS(x_1, \cdot, y_1, q_B).$$
 (5)

for every q_B .

Recall that the view of Bob is defined as his input, his random input, and the transcript of the communication. By Equation (5), for every q_B the communication transcript between Alice and Bob is identically distributed when the inputs of Alice and Bob are x_0, y_1 and when their inputs are x_1, y_1 . In both cases Bob has the same input y_1 and random input q_B , so Bob's view is identically distributed in both cases, namely for every q_B it holds that

$$VIEW_B(x_0, \cdot, y_1, q_B) = VIEW_B(x_1, \cdot, y_1, q_B).$$
(6)

Equation (6) contradicts the correctness requirement from Definition 1, because $f(x_0, y_1) = b \neq c = f(x_1, y_1)$, whereas the identical distributions of Bob's view imply that Bob has the same output distribution in both cases. Thus, we have reached a contradiction, which concludes the proof of the claim.

Claim 1 and Claim 2 complete the proof of Theorem 1.

3.3 The Round Complexity of Secure Computation against Unbounded Malicious Parties

Typically, multiple rounds and probabilism are crucial ingredients of secure computation. As stated in the following corollary, however, two-party secure computation in the unbounded malicious model is an exception.

Corollary 1. If a function f is securely computable in the unbounded malicious model, then it is so computable by a deterministic single-round (actually, single-message) protocol.

Proof. The corollary follows immediately from our proof of Theorem 1 (rather than from its statement). That proof, shows that, if a function f is computable in the unbounded two-party malicious model, then it is so computed by the protocol of Fig. 1, in which only a single message is exchanged (from A to B).

Together with the above corollary, our proof of Theorem 1 (actually, of Claim 2 alone) also immediately implies the following relationship between secure computation in the unbounded honest-but-curious model and in the unbounded malicious one.

Corollary 2. For every two-argument function f, one of the following holds: Either

- 1. f is securely computable deterministically and in one round in the unbounded malicious model; Or
- 2. f is not securely computable in the unbounded honest-but-curious model, even by probabilistic and multi-round protocols.

4 Characterization of Complete Functions

In this section we prove that every function that contains an insecure minor is *complete* for secure computation. That is, every non-trivial function is complete.

We shall consider secure computation in the (computationally) bounded malicious model. That is, the computation is secure provided that the (malicious) parties run in polynomial time. Thus, for the privacy conditions to hold, the appropriate probability distributions are only required to be *indistinguishable* by polynomial time Turing Machines (rather than identical as in the unbounded case of Definition 2). In our proof we also consider the bounded honest-but-curious model. For lack of space we do not give precise definitions of secure computation in the bounded models, definitions which are much more involved and complex than the definitions in the unbounded models. Such definitions can be found, e.g., in [Gol98]. We note that our results hold for all reasonable definitions of secure computation in these model.

4.1 Reductions and Completeness

As usual, the definition of completeness relies on that of a reduction.

Definition 5 (Reductions). Let $f(\cdot, \cdot)$ and $g(\cdot, \cdot)$ be finite functions. We say that the function g reduces to f in the bounded malicious model (respectively, in the bounded honest-but-curious model) if there exists a transformation¹⁶ from

¹⁶ All reductions presented in this paper consist of efficient and uniform transformations.

any protocol securely computing f in the bounded malicious model (respectively, in the bounded honest-but-curious model) to a protocol securely computing g in the same model.

Definition 6 (Completeness). The function $f(\cdot, \cdot)$ is complete for bounded malicious secure computations (respectively, bounded honest-but-curious secure computations) if every finite function $g(\cdot, \cdot)$ reduces to f in the bounded malicious model (respectively, in the bounded honest-but-curious model).

Informally, a function g reduces to a function f if a secure protocol for f can be converted to a secure protocol for g without any additional assumptions. (Even more informally, f is "harder" to compute securely than g.)

4.2 Our Reduction vs. Black-Box Reductions

As mentioned in the introduction, our reductions are not black-box ones, but are natural and very suitable for investigating which assumptions are sufficient for secure computation. In contrast, black-box reductions are not strong enough to establish our main theorem, the all-or-nothing nature of two-party secure computation. For instance, the (non-trivial) OR function is not complete under black-box reductions [Kil99]¹⁷. Hence, black-box reductions do not give any indication regarding which is the minimal assumption necessary for implementing OR securely. On the other hand, using our notions of reductions and completeness, our main theorem implies that the complexity assumptions necessary for implementing OR securely are exactly the same as for all other non-trivial functions.

Let us now emphasize that our reductions and completeness satisfy basic expected properties of such notions.

Lemma 1. Let f and g be finite functions such that g reduces to f in the bounded malicious model of secure computation. Then any assumption sufficient for computing f securely in the bounded malicious model is also sufficient for securely computing g in the same model.

Proof. Consider a protocol (A_f, B_f) that securely computes f under some assumption ASSUM_f. Since g reduces to f, we can apply the transformation from (A_f, B_f) to obtain a protocol (A_g, B_g) such that if (A_f, B_f) securely computes f then (A_g, B_g) securely computes g. Thus, if ASSUM_f holds then (A_g, B_g) securely computes g.

Furthermore, by definition, these reductions are transitive:

¹⁷ Indeed, it is not hard to see that in any protocol that uses a black-box for OR, the party receiving the output can input 0 to the black-box, thus obtaining the other party's input, without any way of being detected. Since this cannot be prevented, any protocol which is unconditionally secure using an OR black-box can be transformed into an unconditionally secure protocol, implying that only trivial functions can be black-box reduced to OR.

Lemma 2. Let f, g, and h be finite functions. If h reduces to g and g reduces to f then h reduces to f.

Lemma 3. Let f and g be any finite functions. If g can be computed securely in the bounded malicious model without any assumptions then g reduces to f in the bounded malicious model.

Proof. Consider a protocol (A_g, B_g) that computes g securely. The transformation from any protocol (A_f, B_f) securely computing f to a protocol securely computing g ignores (A_f, B_f) and outputs (A_g, B_g) .

Remark 1. We stress that our notion of completeness highlights the all-or-nothing nature of secure computation. Furthermore, by [Yao86,GMW87], if factoring is hard or if trapdoor permutations exist, then all finite functions (even the trivial ones!) are complete.

4.3 Main Theorem

Theorem 2. If $f(\cdot, \cdot)$ is a non-trivial function, then f is complete in the bounded malicious model.

Proof Outline. Although we aim towards the bounded malicious model, our proof of Theorem 2 wanders through the bounded honest-but-curious model (more direct proofs seem problematic¹⁸). We first prove that every non-trivial function is complete in the honest-but-curious model. We then use standard techniques of [GMW87] to transform any secure protocol in the bounded honest-but-curious model into a secure protocol in the bounded malicious model. In general this transformation requires some complexity assumptions, however in our case the protocol in the honest-but-curious model implies these assumptions. Thus, combining the above steps, every non-trivial function is complete in the malicious model.

Proof. We start by proving the analogue of Theorem 2 for the honest-but-curious model.

Claim 3. If a function $f(\cdot, \cdot)$ contains an insecure minor, then f is complete in the bounded honest-but-curious model.

Proof. It is proven in [GV87] that OT is complete in the bounded honest-but-curious model. Therefore, to establish our claim it suffices to prove that whenever f contains an insecure minor then OT reduces to f.

Let (A_f, B_f) be a secure protocol computing the function f in the bounded honest-but-curious model. Because the function f contains an insecure minor, there are values x_0, x_1, y_0, y_1, a, b and c such that $b \neq c$, $f(x_0, y_0) = f(x_1, y_0) = a$, $f(x_0, y_1) = b$, and $f(x_1, y_1) = c$.

```
\begin{aligned} & \textbf{Protocol}\;(A_{\text{OT}}, B_{\text{OT}}) \\ & A_{\text{OT}}\text{'s input:}\; \beta_0, \beta_1 \in \{0, 1\} \\ & B_{\text{OT}}\text{'s input:}\; i \in \{0, 1\} \\ & A_{\text{OT}}\text{'s and}\; B_{\text{OT}}\text{'s code:}\; \text{Execute protocol}\; (A_f, B_f) \text{ on input } x_{\beta_0}, y_{\overline{\imath}}. \\ & \text{Denote by } z_0 \text{ the output of } B_f. \\ & \text{Execute protocol}\; (A_f, B_f) \text{ on input } x_{\beta_1}, y_i. \\ & \text{Denote by } z_1 \text{ the output of } B_f. \\ & B_{\text{OT}}\text{'s output:}\; \text{If } z_i = b \text{ then output } 0, \text{ else output } 1. \end{aligned}
```

Fig. 2. A secure protocol in the bounded honest-but-curious model for computing OT from a function f containing an insecure minor with values $x_0, x_1, y_0, y_1, a, b, c$.

In Fig. 2 we describe a protocol $(A_{\text{OT}}, B_{\text{OT}})$ which securely computes OT using this insecure minor and the protocol (A_f, B_f) .

In Protocol $(A_{\text{OT}}, B_{\text{OT}})$ it holds that $z_i = f(x_{\beta_i}, y_1)$, and, thus, $z_i = b$ if $\beta_i = 0$ (and $z_i = c \neq b$ otherwise), implying that the output of B_{OT} is correct. We next argue that the privacy constrains are satisfied for bounded honest-but-curious parties A_{OT} and B_{OT} . First note that the only messages exchanged in $(A_{\text{OT}}, B_{\text{OT}})$ are during the executions of (A_f, B_f) . Since (A_f, B_f) computes f securely, A_f (and thus A_{OT}) does not learn any information about i. Recall that B_f is not allowed to learn any information that is not implied by his input and the output of the function. In the case of OT, this means B_{OT} should not learn any information about $\beta_{\overline{i}}$. However, the only information that A_{OT} sends that depends on $\beta_{\overline{i}}$ are during the execution of (A_f, B_f) on input $(x_{\beta_{\overline{i}}}, y_0)$ and, thus, $z_{\overline{i}} = a$ for both values of $\beta_{\overline{i}}$. By the fact that (A_f, B_f) computes f securely, B_f does not learn any information on $\beta_{\overline{i}}$.

Note that the above protocol is secure only if B_{OT} is honest. Also note that in protocol $(A_{\text{OT}}, B_{\text{OT}})$ it is important that only B_f gets the outputs z_0 and z_1 of (A_f, B_f) . That is, if A_{OT} gets z_0 or z_1 then she can learn B_{OT} 's input for at least one of the possible values of her input (since either $b \neq a$ or $c \neq a$ or both).

Let us now prove an "hybrid result" bridging the completeness in the two bounded models of secure computation.

Claim 4. Let $f(\cdot, \cdot)$ be any finite function. If f is complete in the bounded honest-but-curious model then it is complete in the bounded malicious model.

Proof Sketch. Let g be any finite function. We need to prove that g reduces to f in the bounded malicious model. We are promised that g reduces to f in the bounded honest-but-curious model. That is, there is a transformation from any protocol securely computing f to one securely computing g in the bounded honest-but-curious model.

To obtain a protocol securely computing g in the malicious model, we proceed as follows. First, there exists a transformation mapping any protocol that

¹⁸ For example, we cannot use Kilian's reduction [Kil91] from OT to a two-sided computation of OR in the bounded malicious model.

securely computes OT in the bounded honest-but-curious model into a one-way function [IL89]. Second, since f is complete in the bounded honest-but-curious model, this implies that there exists a transformation mapping any protocol that securely computes f in the bounded honest-but-curious model into a one-way function. Third, one-way functions imply pseudo-random generators [HILL91], which in turn imply bit commitment [Nao89]. Finally, bit commitment implies that it is possible to transform any protocol securely computing an arbitrary function g in the bounded honest-but-curious model into a protocol securely computing g in the bounded malicious model [GMW87]. Putting the above together, we obtain a transformation from a protocol securely computing f in the bounded honest-but-curious model to one computing the function g in the bounded malicious model.

We are ready to complete the proof of Theorem 2. By Theorem 1 any non-trivial function f contains an insecure minor. Thus, by Claim 3 and Claim 4, f is complete in the bounded malicious model.

Acknowledgments

We thank Joe Kilian and Eyal Kushilevitz for insightful comments. We also thank Lior Pachter for risking his life to save a preliminary version of this work.

References

- Blu82. M. Blum. Coin flipping by phone. *IEEE Spring COMPCOM*, pages 133–137, 1982.
- BGW88. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *Proc. of the 20th Symp. on Theory of Computing*, pages 1–10, 1988.
- CCD88. D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In Proc. of the 20th Symp. on Theory of Comp., pages 11–19, 1988
- CK88. C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In Proc. of the 29th IEEE Symp. on Foundations of Computer Science, pages 42–52, 1988.
- CK89. B. Chor and E. Kushilevitz. A zero-one law for Boolean privacy. SIAM J. on Discrete Math., 4(1):36–47, 1991. Prelim. version in STOC '89, 1989.
- Cré88. C. Crépeau. Equivalence between two flavors of oblivious transfers. In Advances in Cryptology CRYPTO '87, volume 293 of Lecture Notes in Computer Science, pages 350–354. Springer, 1988.
- EGL85. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *CACM*, 28(6):637–647, 1985.
- FMR84. M. J. Fischer, S. Micali, and C. Rackoff. A secure protocol for the oblivious transfer. Presented in EUROCRYPT '84, 1984. Printed version in *J. of Cryptology*, 9(3):191–195, 1996.
- GMR85. S. Goldwasser, M. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM J. on Computing, 18:186–208, 1989. Preliminary version in STOC '85, 1985.
- GMW86. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or all languages in NP have zero-knowledge proof systems. In *J. ACM*, 38(1):691–729, 1991. Preliminary version in *FOCS '86*, 1986.

- GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th Symp. on the Theory of Comp.*, pages 218–229, 1987.
- Gol98. O. Goldreich. Secure multi-party computation (working draft). Available from http://www.wisdom.weizmann.ac.il/~oded/foc.html, 1998.
- GV87. O. Goldreich and R. Vainish. How to solve any protocol problem—an efficiency improvement. In *Advances in Cryptology CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 1988.
- HILL91. J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. Technical Report TR-91-068, International Computer Science Institute. 1991.
- IL89. R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proc. of the 30th IEEE Symp. on Foundations of Computer Science*, pages 230–235, 1989.
- IR89. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proc. of the 21st ACM Symp. on the Theory of Computing*, pages 44–61, 1989.
- Kil88. J. Kilian. Basing cryptography on oblivious transfer. In *Proc. of the 20th ACM Symp. on the Theory of Computing*, pages 20–31, 1988.
- Kil90. J. Kilian. Uses of Randomness in Algorithms and Protocols. MIT Press, 1990.
- Kil91. J. Kilian. A general completeness theorem for two-party games. In *Proc.* of the 23th ACM Symp. on the Theory of Computing, pages 553–560, 1991.
- Kil99. J. Kilian. Personal communication. 1999.
- KKMO98. J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. 1998. To appear in SIAM J. on Computing. This is the journal version of [Kil91,KMO94].
- KMO94. E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in multi-party private computations. In *Proc. of the 35th IEEE Symp. on Foundations of Computer Science*, pages 478–491, 1994.
- Kus89. E. Kushilevitz. Privacy and communication complexity. SIAM J. on Discrete Mathematics, 5(2):273–284, 1992. Preliminary version in FOCS '89, 1989.
- MR92. S. Micali and P. Rogaway. Secure computation. In Advances in Cryptology
 CRYPTO '91, vol. 576 of Lecture Notes in Computer Science, pages 392—
 404. Springer-Verlag, 1992. An updated version presented at: Workshop on
 Multi-Party Secure Computation. Weizmann Inst., Israel, June 1998.
- Nao89. M. Naor. Bit commitment using pseudorandom generators. J. of Cryptology, 4:151–158, 1991. Preliminary version in Advances in Cryptology CRYPTO '89, 1989.
- Rab81. M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
- Yao82. A. C. Yao. Protocols for secure computations. In *Proc. of the 23th IEEE Symp. on Foundations of Computer Science*, pages 160–164, 1982.
- Yao86. A. C. Yao. How to generate and exchange secrets. In Proc. of the 27th IEEE Symp. on Foundations of Computer Science, pages 162–167, 1986.

Adaptive Security for Threshold Cryptosystems

Ran Canetti¹, Rosario Gennaro¹, Stanisław Jarecki², Hugo Krawczyk³, and Tal Rabin¹

¹ IBM T.J.Watson Research Center PO Box 704, Yorktown Heights, NY 10598, USA {canetti,rosario,talr}@watson.ibm.com ² MIT Laboratory for Computer Science 545 Tech Square, Cambridge, MA 02139, USA stasio@theory.lcs.mit.edu ³ Department of Electrical Engineering, Technion Haifa 32000, Israel and IBM T.J. Watson Research Center, New York, USA hugo@ee.technion.ac.il

Abstract. We present adaptively-secure efficient solutions to several central problems in the area of threshold cryptography. We prove these solutions to withstand adaptive attackers that choose parties for corruption at any time during the run of the protocol. In contrast, all previously known efficient protocols for these problems were proven secure only against less realistic static adversaries that choose and fix the subset of corrupted parties before the start of the protocol run. Specifically, we provide adaptively-secure solutions for distributed key generation in discrete-log based cryptosystems, and for the problem of distributed generation of DSS signatures (threshold DSS). We also show how to transform existent static solutions for threshold RSA and proactive schemes to withstand the stronger adaptive attackers. In doing so, we introduce several techniques for the design and analysis of adaptively-secure protocols that may well find further applications.

1 Introduction

Distributed cryptography has received a lot of attention in modern cryptographic research. It covers a variety of areas and applications, from the study of secret-sharing schemes, to the distributed computation of general functions using secure multi-party protocols, to the design and analysis of specific threshold cryptosystems. Two main goals that motivate this research area are: (i) provide security to applications that are inherently distributed, namely, several parties are trying to accomplish some common task (e.g., secure elections, auctions, games) in the presence of an attacker, and (ii) avoid single points-of-failure in a security system by distributing the crucial security resources (e.g. sharing the ability to generate signatures). In both cases the underlying assumption is that an attacker can penetrate and control a portion of the parties in the system but not all of them.

Coming up with correct protocols for meeting the above goals has proven to be a challenging task; trying to design protocols that are practical as well as fully-analyzed is even more challenging. One inherent difficulty in the analysis of cryptographic systems, in general, is the need to define a mathematical model that is strong enough to capture realistic security threats and attacker's capabilities, and which, at the same time, allows to prove the security of sound solutions. Due to the complexity of building and reasoning about distributed protocols, this difficulty is even greater in the area of distributed cryptography. In this case a large variety of security models have been proposed. This is not just due to "philosophical disagreements" on what the best way to model reasonable attackers is, but it is also influenced by our ability (or lack of it) to prove security of protocols under strong adversarial models.

One major distinction between security models for distributed protocols is whether the attacker is *static* or *adaptive* (the latter is also called "dynamic"). In both cases the attacker is allowed to corrupt any subset of parties up to some size (or threshold) as specified by the security model. However, in the case of an adaptive adversary the decision of which parties to corrupt can be made at any time during the run of the protocol and, in particular, it can be based on the information gathered by the attacker during this run. In contrast, in the static case the attacker must choose its victims independently of the information it learns during the protocol. Therefore, the subset of corrupted parties can be seen as chosen and fixed by the attacker before the start of the protocol's run.

While the adaptive attacker model appears to better capture real threats, the bulk of published works on distributed cryptography assumes a static adversary. This is due to the difficulties encountered when trying to design and prove protocols resistant to adaptive adversaries. Still, general constructions have been shown in the adaptive-adversary model for secure distributed evaluation of any polynomial-time computable function (see below). However, these general results do not provide sufficiently efficient and practical solutions. Until now, no efficient adaptive solutions for threshold cryptosystems were known.

Our Contribution. The main contribution of this paper is in providing concrete, fully-specified, fully-analyzed solutions to some of the central problems in threshold cryptography, and proving their security in the adaptive adversary model. Our solutions add little overhead relative to existing solutions for the same problems in the static-adversary model. They are also constant-round; namely, the number of rounds of communication is fixed and independent of the number of parties in the system, the input length, or the security parameter. Thus we believe that our protocols can be of real use in emerging threshold cryptography applications. Very importantly, we provide full analysis and proofs for our solutions. This is essential in an area where simple intuition is usually highly misleading and likely to produce flawed protocols.

In addition, our work introduces general new techniques for the design and analysis of protocols in the adaptive-adversary model. In section 2 we give an overview of these techniques to aid the understanding of the protocols and the proofs given in the following sections. We also hope that these techniques will be applicable to the design of adaptively-secure solutions to other problems.

In section 3 we start the presentation of our protocols with an adaptively-secure solution for the distributed generation of keys for DSS, El Gamal, and other discrete-log based public-key systems (for signatures and encryption). Such a protocol is not only needed for generating keys in a distributed way without a trusted party, but it is also a crucial component of many other cryptographic protocols. We illustrate this in our own work by using it for distributed generation of the $r = g^k$ part in DSS signatures.

In section 4 we present a threshold DSS protocol for the shared generation of DSS signatures that withstands adaptive attackers. We first show a simplified protocol where the attacker can control up to t < n/4 players. This protocols helps in highlighting and understanding many of our new and basic techniques. Next we describe an optimal-resilience, t < n/2, threshold DSS protocol.

In section 5 we show how our techniques can be used to achieve adaptively-secure distributed protocols for other public-key systems. We show how to modify existing threshold RSA protocols (proven in the static model) to obtain adaptively-secure threshold RSA. Here also we achieve optimal-resiliency and constant-round protocols. Similarly, our techniques allow for "upgrading" existent proactive discrete-log based threshold schemes from the static-adversary model to the adaptive one.

Related Work. Our work builds directly on previous protocols that were secure only in the static-adversary model, particularly on [Ped91b,GJKR99] for the distributed key generation and [GJKR96] for the threshold DSS signatures. We modify and strengthen these protocols to achieve adaptive security. Similarly, our solution to adaptive threshold RSA is based on the previous work of [Rab98].

As said before, if one is not concerned with the practicality of the solutions then general protocols for secure distributed computation of polynomial-time functions in the presence of adaptive adversaries are known. This was shown in [BGW88,CCD88] assuming (ideal) private channels. Later, [BH92] showed that with the help of standard encryption and careful erasure of keys one can translate these protocols into the model of public tappable channels. Recently, [CFGN96] showed how to do this translation without recurring to erasures but at the expense of a significant added complexity. Other recent work on the adaptive-adversary model includes [Can98,CDD+99]. Also, independently from our work, adaptively-secure distributed cryptosystems have been recently studied in [FMY].

2 Technical Overview: Road-Map to Adaptive Security

This section provides an overview of some basic technical elements in our work. It is intended as a high-level introduction to some of the issues that underly the protocol design and proofs presented in this paper (without getting into a detailed description of the protocols themselves). We point out to some general aspects of our design and proof methodology, focusing on the elements that are essential to the treatment of the adaptive-adversary model in general, and to the understanding of our new techniques. For simplicity and concreteness, our

presentation focuses mainly on threshold signature schemes which are threshold solutions to some existing centralized scheme like DSS or RSA. However, most of the issues we raise are applicable to other threshold functions and cryptosystems.

Threshold Signature Schemes. A threshold signature scheme consists of a distributed key generation protocol, a distributed signature generation protocol and a centralized verification algorithm. The signing servers first run the key generation, and obtain their private key shares and the global public key. Next, whenever a message needs to be signed, the servers invoke the signature generation protocol. The definition of secure threshold signature scheme makes two requirements. The first is unforgeability, which says, roughly, that, even after interacting with the signing parties in the initial generation of the distributed key and then in the signing protocol invoked on adaptively chosen messages, the adversary should be unable (except with negligible probability) to come up with a message that was not signed by the servers together with a valid signature. The second requirement is robustness, which says that whenever the servers wish to sign a message, a valid signature should be generated. Our model and definitions are a natural adaptation to the adaptive-adversary model of the definitions for threshold signatures found in [GJKR96], which in turn are based on the unforgeability notions of [GMR88].

Proofs by Reduction and the Central Role of Simulations. We use the usual "reductionist" approach for the proofs of our protocols. Namely, given an adversary A that forges signatures in the distributed setting, we construct a forger \mathcal{F} that forges signatures of the underlying centralized signature scheme. Thus, under the assumption that the centralized scheme is secure, the threshold signature scheme must also be secure. A key ingredient in the reduction is a "simulation" of the view of the adversary A in its run against the distributed protocol. That is, the forger \mathcal{F} builds a virtual distributed environment where the instructions of \mathcal{A} are carried out. Typically, first \mathcal{F} has to simulate to \mathcal{A} an execution of the distributed key generation that results in the same public key against which \mathcal{F} stages a forgery. Then \mathcal{A} will successively invoke the distributed signature protocol on messages of his choice, and \mathcal{F} , having access to a signing oracle of the underlying centralized signature scheme, has to simulate to A its view of an execution of the distributed signature protocol on these messages. Eventually, if \mathcal{A} outputs a forgery in the imitated environment, \mathcal{F} will output a forgery against the centralized signature scheme. Two crucial steps in the analysis of this forger are: (1) Demonstrate that the adversary's view of the simulated interaction is indistinguishable from its view of a real interaction with parties running the threshold scheme. (2) Demonstrate that the forger can translate a successful forgery by the adversary (in the simulated run) to a successful forgery of the centralized signature scheme.

The first step is the technical core of our proofs. Furthermore, to carry out the second step we need simulators that are able to generate views that are indistinguishable from the view of the adversary under a given conditioning. More specifically, \mathcal{F} has to simulate a run of the distributed key generation that arrives at a given public key of the underlying centralized scheme, and it has

to simulate a run of the distributed signature scheme that arrives at a *given* signature output by the signing oracle of the centralized scheme. We refer to this as the problem of *hitting* a particular value in the simulation.

Problems with Simulating an Adaptive Adversary. The above proof technique is also used in the case of static adversaries. However, an adaptive adversary can corrupt any player at any time (as long as not too many parties are corrupted) and at that point the simulator needs to be able to provide the attacker with the current internal state of the broken party. In particular, this information must be consistent with the information previously seen by the attacker. Providing this information is typically the main difficulty in proving adaptive security of protocols.

We demonstrate this difficulty with a simplified example adapted from our protocols: Assume that the protocol specifies that each server P_i chooses a random exponent x_i and makes public (broadcasts) g^{x_i} , where g is a known group generator. Next, the adversary sees $g^{x_1}...g^{x_n}$, corrupts, say, 1/3 of the servers, and expects to see the secret exponents x_i of the corrupted servers being consistent with g^{x_i} . Since the simulator cannot predict which servers will be corrupted, the only known way to carry out the simulation is to make sure that the simulator knows in advance the secret values x_i of all the servers, even though the adversary corrupts only a fraction of them. However, this is not possible in our protocols where the x_i 's encode a secret quantity x (such as a signing key) unknown to the simulator. Note that trying to guess which servers will be corrupted does not help the simulation: there is an exponential number of such sets. (In contrast, the simulation of such a protocol is possible in the case of a static attacker where the simulator knows in advance the set of corrupted players.)

Erasures. One way to get around the above problem in the adaptive-adversary model is to specify in the protocol that the servers erase the private values x_i before the values g^{x_i} are broadcasted. Now, when corrupting P_i , this information is not available to the adversary in the real run of the protocol and therefore there is no need to provide it in the simulation. However, this technique can only be applied when the protocol no longer needs x_i . A careful use of erasures is at the core of the design of our protocols. In some cases, this requires that information that could have been useful for the continuation of the protocol be erased. Two examples of crucial erasures in our protocols are the erasure of some of the verification information kept by Pedersen's VSS protocol [Ped91a] (which we compensate for with the use zero-knowledge proofs – see below), and the erasure of all temporary information generated during each execution of the signature protocol. Furthermore, erasures simplify the task of implementing private channels with conventional encryption in the adaptive model (see below).

¹ Successful erasure of data is not a trivial task; one needs to make sure that the data, and all its back-ups, are carefully overwritten. In our setting, we trust *uncorrupted* parties to properly erase data whenever required by the protocol. (See [CFGN96,Can98] for further discussion on the issue of data erasures.)

Rewinding the Adversary. Another useful technique for getting around the problem of having to present the attacker with information not available to the simulator is rewinding. This is a well-known technique for proving zero-knowledge (and other) protocols. In its essence it allows a simulator that is "in trouble" with its simulation to rewind the adversary's state to a previous computation state, and restart the computation from there. At this point the simulation will try some other random choices in the computation hoping that it will not end in another bad position as before. Thus, in the case where the chances to get "stuck" again are not too large then rewinding is a very useful technique. In a second try, after the simulator makes some different choices, the adversary will hopefully not hit an "inconsistent" situation again.

Note that rewinding is a proof technique, not a protocol action. Yet, one has to design the protocol in ways that make rewinding a useful tool. As in the case of erasures, correct use of rewinding requires care. Following is an important instance where an improper use of rewinding can render the whole simulation useless.² Assume that the distributed adversary \mathcal{A} asks for signatures on a sequence of messages m_1, m_2, \ldots For each message m_i , in order to simulate the signature protocol, the forger \mathcal{F} must ask the signing oracle of the underlying centralized signature scheme for a signature on m_i . Assume now that during the k^{th} signature protocol (while simulating the distributed signing of message m_k) the simulator gets "stuck" and needs to rewind the adversary back to a previous signature protocol, say the j^{th} one for j < k. Now the adversary is rewinded back to m_i , the subsequent view is different, so with all likelihood the adversary will start asking for signatures on different messages. In other words the sequence of messages is now $m_1, \ldots, m_j, m'_{i+1}, \ldots, m'_k, \ldots$ However the sequence of messages asked by the forger $\mathcal F$ to the signature oracle is $m_1, \ldots, m_j, m_{j+1}, \ldots, m_k, m'_{j+1}, \ldots, m'_k$, i.e. the forger has now asked more messages than the adversary (indeed since the adversary was rewinded he has no recollection of asking the messages m_{i+1}, \ldots, m_k). This means that the adversary \mathcal{A} may output one of those messages as a successful forgery, but such event will not count as a success for the forger \mathcal{F} .

It is important then to confine rewinding of the adversary *inside* a simulation of a single run of the signature protocol. One of the tools that we use to ensure that our protocols are simulatable in such a way is the erasure of local temporary information generated by our protocols.

The Single-Inconsistent-Player Technique. Another concern that needs to be addressed is making sure that rewindings do not take place too often (otherwise the simulator may not run in polynomial time). Very roughly, we guarantee this property as follows. We make sure that the simulator can, at any point of the simulated run, present the attacker with a correct internal state (i.e. state that is consistent with the attacker's view) for all honest players except, maybe,

² We remark that the rewinding technique may also cause difficulties when the signature protocol is composed with other protocols, and in particular when several copies of the signature protocol are allowed to run *concurrently*. We leave these issues out of the scope of this work. See more details in [DNS98,KPR98].

for one server. The identity of this server is chosen at random. Moreover, the view of the attacker is independent from this choice. This guarantees that the inconsistent server is corrupted (and then the simulation is rewinded) with probability at most one half (this probability is given by the ratio t/n of corrupted players). Thus, the expected number of rewindings is at most one.

Zero-Knowledge to the Rescue, and Efficiently. Another tool used in our protocols are zero-knowledge proofs [GMR89], and more specifically zero-knowledge proofs of knowledge. Useful as they are, zero-knowledge proofs may add significant complexity to the protocols, degrade performance, and increase communication. We show how to make intensive use of zero-knowledge proofs with significant savings in complexity. Specifically, we show how to achieve the effect of $O(n^2)$ zero-knowledge proofs of knowledge (where each of the n players proves something to each of the other players) in a *single* 3-move *honest verifier* zero-knowledge proof. This is done by implementing the honest verifier using a distributed generation of a challenge by all players. We implement this technique for Schnorr's proof of possession of discrete-log [Sch91]; the same technique can be used for other zero-knowledge protocols as well [CD98].

Maintaining Private Channels in the Adaptive Model. An important observation about the design of our protocols is that we specify them using the abstraction of "private channels" between each pair of parties. This is a usual simplifying approach in the design of cryptographic protocols: The underlying assumption is that these private channels can be implemented via encryption. In the case of adaptive security, however, this simple paradigm needs to be re-examined. Indeed, a straightforward replacement of private channels with encryption could ruin the adaptive security of our protocols. Fortunately, in settings where data erasures are acceptable (and in particular in our setting) a very simple technique exists [BH92] for solving this problem. It involves local refreshment of (symmetric) encryption keys by each party, using a simple pseudorandom generator and without need for interaction between the parties. This adds virtually no overhead to the protocols beyond the cost of symmetric encryption itself.

3 Adaptively-Secure Distributed Key Generation

A basic component of threshold cryptosystems based on the difficulty of computing discrete logarithms is the shared generation of a secret x for which the value g^x is made public. Not only is this needed to generate a shared key without a trusted dealer but it is also a sub-module of other protocols, e.g. as used in our own threshold DSS scheme for generating $r = g^{k^{-1}}$ (see Sec. 4). We call this module a "Distributed Key Generation" (DKG) protocol. The primitive of

³ Recall that 3-move zero-knowledge proofs cannot exist for *cheating* verifiers if the underlying problem is not in BPP [GK96,IS93]. Thus, the distributed nature of the verifier in our implementation is essential for "forcing honesty". In particular, our simulation of these proofs does not require rewinding at all.

Distributed Key Generation for Discrete–Log Based Cryptosystems is defined in [GJKR99], yet the solution provided in that paper is proven secure only against a non-adaptive adversary. Briefly stating, a DKG protocol is performed by n players $P_1, ..., P_n$ on public input (p, q, g) where p and q are large primes, q divides p-1, and q is an element of order q in Z_p^* . DKG generates a Shamir secret sharing of a uniformly distributed random secret key $x \in Z_q$, and makes public the value $q = q^x \mod p$. At the end of the protocol each player has a private output q, called a share of q. The protocol is secure with threshold q, if in the presence of an adversary who corrupts at most q players, the protocol generates the desired outputs and does not reveal any information about q except for what is implied by the public value q mod q.

To ensure the ability to use the DKG protocol as a module in a larger, adaptively secure, threshold scheme (like the DSS-ts signature scheme of Section 4) we add to the requirements of [GJKR99] that at the end of the protocol each party must erase all the generated internal data pertaining to this execution of the protocol except of course for its private output x_i . This requirement ensures that the simulator of the threshold scheme within which DKG is used as a module (e.g. DSS-ts) is able to produce the internal state of a player whom the adversary corrupts after the execution of the DKG module is completed.

Distributed Key Generation Protocol. We present a distributed key generation protocol DKG with resilience t < n/3, which is simple to explain and already contains the design and analysis ideas in our work. (See below for the modifications required to achieve optimal resilience t < n/2.) DKG is based on the distributed key generation protocol of [GJKR99] which is proven secure only against a non-adaptive adversary. (Some of the changes made to this protocol in order to achieve adaptive security are discussed below.)

Protocol DKG presented in Fig.1, starts with inputs (p, q, g, h) where (p, q, g) is a discrete-log instance and h is a random element in the subgroup of Z_p^* generated by g. When DKG is executed on inputs (p, q, g) only, a random h must first be publicly generated as follows: If q^2 does not divide p-1, the players generate a random $r \in Z_p^*$ via a collective coin-flipping protocol [BGW88] and take $h = r^{(p-1)/q} \mod p$. The protocol proceeds as follows: The first part of generating x is achieved by having each player commit to a random value z_i via a Pedersen's VSS [Ped91a,Ped91b,GJKR99]. These commitments are verified by the other players and the set of parties passing verification is denoted by QUAL. Then the shared secret x is set (implicitly) to $x = \sum_{i \in QUAL} z_i \mod q$. We denote this subprotocol by Joint-RVSS (see Figure 2). In addition to enabling the generation of a random, uniformly distributed value x, Joint-RVSS has the side effect of having each player P_i broadcast an information-theoretically private commitment to z_i of the form $C_{i0} = g^{z_i} h^{f_i'(0)} \mod p$, where f_i' is a random-

⁴ We chose to write DKG with h as an input so that it could be invoked as a module by the DSS-ts scheme of Section 4 without generating h each time.

⁵ See [GJKR99] for an analysis of Joint-RVSS in the non-adaptive adversarial model. The same analysis applies to the adaptive model treated here.

Input: Parameters (p, q, g), and h an element in the subgroup generated by g. **Public Output:** y the public key

Secret Output of P_i : x_i the share of the random secret x

(all other secret outputs are erased)

Other Public Output: Public commitments.

Generating x:

Players execute Joint-RVSS(t, n, t)

- 1. Player P_i gets the following secret outputs of Joint-RVSS:
 - $-x_i, x_i'$ his share of the secret and the associated random value (resp.)
 - $-f_i(z), f_i'(z)$ polynomials he used to share his contribution $z_i = f_i(0)$ to x.
 - $-s_{ji}, s'_{ji}$ for j = 1..n the shares and randomness he received from others Players also get public outputs C_{ik} for i = 1..n, k = 0..t and the set QUAL.

Extracting $y = g^x \mod p$:

Each player exposes $y_i = g^{z_i} \mod p$ to enable the computation of $y = g^x \mod p$.

- 2. Each player P_i , $i \in QUAL$, broadcasts $A_i = g^{f_i(0)} = g^{z_i} \mod p$ and $B_i = h^{f'_i(0)} \mod p$, s.t. $C_{i0} = A_i B_i$. P_i also chooses random values r_i and r'_i and broadcasts $T_i = g^{r_i}, T'_i = h^{r'_i} \mod p$.
- 3. Players execute $\mathsf{Joint}\text{-}\mathsf{RVSS}(t,n,t)$ for a joint random challenge d. Player P_i sets his local share of the secret challenge to d_i . All other secret output generated by this $\mathsf{Joint}\text{-}\mathsf{RVSS}$ and held by P_i is erased.
- 4. Each player broadcast d_i . Set $d = \mathsf{EC\text{-Interpolate}}(d_1, \dots, d_n)$.
- 5. P_i broadcasts $R_i = r_i + d \cdot f_i(0)$ and $R'_i = r'_i + d \cdot f'_i(0)$
- 6. Player P_j checks for each P_i that $g^{R_i} = T_i \cdot A_i^d$ and $h^{R'_i} = T'_i \cdot B_i^d$. If the equation is not satisfied then P_j complains against P_i .
- 7. If player P_i receives more than t complaints, then P_j broadcasts s_{ij} . Set $z_i = \mathsf{EC-Interpolate}(s_{i1},\ldots,s_{in})$ and $A_i = g^{z_i}$.
- 8. The public value y is set to $y = \prod_{i \in OUAL} A_i \mod p$.
- 9. Player P_i erases all secret information aside from his share x_i .

Fig. 1. DKG - Distributed Key Generation, $n \geq 3t + 1$

izing polynomial chosen by P_i in the run of Joint-RVSS. We will utilize these commitments to "extract" and publish the public key $y = g^x \mod p$.

We have that $g^x = g^{\sum_{i \in QUAL} z_i} = \prod_{i \in QUAL} g^{z_i} \mod p$. Thus, if we could have each player "deliver" g^{z_i} in a verifiable way then we could compute y. To that end, we require P_i to "split" his commitment C_{i0} into the two components $A_i = g^{z_i}$ and $B_i = h^{f_i'(0)}$ (Step 2). To ensure that he gives the correct split, P_i proves that he knows both $Dlog_g A_i$ and $Dlog_h B_i$ with Schnorr's 3-round zero-knowledge proof of knowledge of discrete-log [Sch91]. We exploit the fact that each player is proving his statement to many verifiers by generating a single challenge for all these proofs. Joint challenge d is generated with Joint-RVSS with public reconstruction (Steps 3-4). The proof completes in Steps 5-6. For the interesting properties of this form of zero-knowledge proof see Section 2. If a player fails to prove a correct split, then his value z_i is reconstructed using polynomial interpolation with the error-correcting code procedure such as [BW],

Threshold Parameters: (t, n, t')

Input: Parameters (p, q, g) and element h generated by g

Public Output: C_{ik} for i = 1..n, k = 0..t' (referred to as the "commitment to the polynomial"). Set QUAL of non-disqualified players

Secret Output of P_i : x_i and x'_i the share and the associated random value $f_i(z), f'_i(z)$ the polynomials used to share z_i s_{ji}, s'_{ji} for j = 1..n shares received from P_j

- 1. Each player P_i performs a Pedersen-VSS of a random value z_i as a dealer:
 - (a) P_i chooses two random polynomials $f_i(z), f_i'(z)$ over Z_q of degree t': $f_i(z) = a_{i0} + a_{i1}z + ... + a_{it'}z^{t'}$ $f_i'(z) = b_{i0} + b_{i1}z + ... + b_{it'}z^{t'}$ Let $z_i = a_{i0} = f_i(0)$. P_i broadcasts $C_{ik} = g^{a_{ik}}h^{b_{ik}} \mod p$ for k = 0, ..., t'. P_i sends shares $s_{ij} = f_i(j), s_{ij}' = f_i'(j) \mod q$ to each P_j for j = 1, ..., n.
 - (b) Each P_i verifies the shares received from other players for i = 1, ..., n

$$g^{s_{ij}} h^{s'_{ij}} = \prod_{k=0}^{t'} (C_{ik})^{j^k} \bmod p$$
 (1)

If the check fails for an index i, P_i broadcasts a *complaint* against P_i .

- (c) Each player P_i who, as a dealer, received a complaint from player P_j broadcasts the values s_{ij}, s'_{ij} that satisfy Eq. (1).
- (d) Each player builds the set of players *QUAL* which excludes any player
 - who received more than t complaints in Step 1b, or
 - answered to a complaint in Step 1c with values that violate Eq.(1).
- 2. The shared random value x is not computed by any party, but it equals $x = \sum_{i \in QUAL} z_i \mod q$. Each P_i sets his share of the secret to $x_i = \sum_{j \in QUAL} s_{ji} \mod q$ and the associated random value $x_i' = \sum_{j \in QUAL} s_{ji}' \mod q$.

Fig. 2. Joint-RVSS - Joint Pedersen VSS, $n \ge 2t + 1$

which we denote by EC-Interpolate (Steps 4, 7). The players then *erase* all the information generated during the protocol except of their share of the secret key.

Proving Adaptive Security of Key Generation. In Figure 3 we present a simulator SIM-DKG for the DKG protocol.⁶ This simulation is the crux of the proof of secrecy in the protocol, namely, that nothing is revealed by the protocol beyond the value $y = g^x \mod p$ (to show this we provide the value of y as input to the simulator and require it to simulate a run of the DKG protocol that ends with y as its public output). We denote by \mathcal{G} (resp. \mathcal{B}) the set of currently good (resp. bad) players. The simulator executes the protocol for all the players in \mathcal{G} except one. The state of the special player P (selected at random) is used by the simulator to "fix" the output of the simulation to y, the required public key. Since the simulator does not know $Dlog_q y$ it does not know some secret

⁶ If DKG is preceded with generation of h (see above), the simulator plays the part of the honest players in that protocol before running SIM-DKG to simulate DKG.

information relative to this special player (in particular the component z_P that this player contributes to the secret key). This lack of knowledge does not disable the simulator from proving that it knows P's contribution (Steps 2-6 in DKG) since the simulator can utilize a simulation of this ZK proof. However, if the adversary corrupts P during the simulation (which happens with probability < 1/2) the simulator will not be able to provide the internal state of this player. Thus, the simulator will need to rewind the adversary and select another special player P'. The simulation will conclude in expected polynomial time.

Input: public key y and parameters (p, q, g) and h an element generated by

1. Perform Step 1 of DKG on behalf of the players in \mathcal{G} . At the end of this step the set QUAL is defined. SIM-DKG knows all polynomials $f_i(z)$, $f'_i(z)$ for $i \in QUAL$ (as it controls a majority of the players). In particular, SIM-DKG knows the values $f_i(0), f'_i(0)$.

Perform the following pre-computations:

- Choose at random one uncorrupted player $P \in \mathcal{G}$
- Compute $A_i = g^{f_i(0)}, B_i = h^{f_i'(0)}$ for $i \in QUAL \setminus \{P\}$ Set $A_P^* = y \cdot \prod_{i \in (QUAL \setminus \{P\})} (A_i)^{-1} \mod p$, and $B_P^* = C_{P0}/A_P^* \mod p$
- Pick values $d, R_P, R_P' \in_R Z_q$, set $T_P^* = g^{R_P} \cdot (A_P^*)^{-d}$ and $T_P'^* = h^{R_P'} \cdot (B_P^*)^{-d}$
 - 2. For each player $i \in \mathcal{G} \setminus \{\mathcal{P}\}$ execute Step 2 according to the protocol. For player P broadcast $A_P^*, B_P^*, T_P^*, T_P^{'*}$ which were computed previously.
 - 3. Perform the Joint-RVSS(t, n, t) protocol on behalf of the uncorrupted players. Note that SIM-DKG knows the shares d_i , $i \in \mathcal{B}$. Erase all the secret output of the uncorrupted players in this protocol. Pick a t-degree polynomial $f_d^*(z)$ s.t. $f_d^*(0) = d$ and $f_d^*(i) = d_i$ for $i \in \mathcal{B}$. Set $d_i^* = f_d^*(i)$ for $i \in \mathcal{G}$.
 - 4. Broadcast d_i^* for each $i \in \mathcal{G}$.
 - 5. Broadcast $R_i = r_i + d \cdot f_i(0) \mod q$ and $R'_i = r'_i + d \cdot f'_i(0) \mod q$ for $i \in \mathcal{G} \setminus \{P\}$ and $R_P^*, R_P^{'*}$ for player P.
- 6. Execute Step 6 of DKG for all players. (Notice that the corrupted players can publish a valid complaint only against one another.)
- 7. For each player with more than t complaints participate in the reconstruction of their value. Note that only players in \mathcal{B} can have more than t complaints.
- 8. Erase all information aside from the value x_i .

Fig. 3. SIM-DKG - Simulator for the Distributed Key Generation Protocol DKG

Lemma 1. Simulator SIM-DKG on input (p,q,q,h,y) ends in expected polynomial time and computes a view for the adversary that is indistinguishable from a view of the protocol DKG on input (p, q, g, h) and output y.

Proof: First we show that SIM-DKG outputs a probability distribution which is identical to the distribution the adversary sees in an execution of DKG that produces y as an output. In the following denote by G_g the subgroup of Z_p^* generated by g.

- 1. The first step is carried out according to the protocol, thus values $f_i(j), f'_i(j), i \in \mathcal{G}, j \in \mathcal{B}$, and $C_{ik}, i \in \mathcal{G}, k = 0 \dots t$ have the required distribution.
- 2. The values A_i for $i \in \mathcal{G} \setminus \{P\}$ are distributed exactly as in the real protocol. The value $A_P^* = y \cdot \prod_{i \in (QUAL \setminus \{P\})} (A_i)^{-1} = y \cdot \prod_{i \in (\mathcal{G} \setminus \{P\})} (A_i)^{-1} \cdot \prod_{i \in (\mathcal{B} \cap QUAL)} (A_i)^{-1}$ is distributed uniformly in G_g and independently from the values A_i for all i. This is because y is random and uniformly distributed and the A_i for $i \in \mathcal{B}$ are generated independently from the other ones (because Joint-RVSS is information theoretically private).

A similar argument holds for values B_i , $i \in \mathcal{G} \setminus \{P\}$ and B_P^* . Finally, values T_i, T_i' , $i \in \mathcal{G} \setminus \{P\}$ are picked at random in G_g as in the protocol. Values $T_P^*, T_P^{'*}$ are also uniformly distributed in G_g . Thus the view in this step is identical.

- 3. Here SIM-DKG performs a Joint-RVSS(t, n, t) protocol on behalf of the players in \mathcal{G} exactly as in the protocol. Thus the view in this step is identical.
- 4. In this step SIM-DKG broadcasts shares d_i^* of a new polynomial $f_d^*(z)$ which is random subject to the constraint that $f_d^*(j) = f_d(j)$ for $j \in \mathcal{B}$ and $f_d^*(0) = d$. Although these d_i^* 's are not the same values held as shares by the players in the previous step, the view for the adversary is still the same as in the real protocol. This is because the adversary has seen a Joint-RVSS of value d' and at most t points of the sharing polynomial f. It is easily seen that for any other value d there is another polynomial that passes through the t points held by the adversary and the free term d. Notice that since only the d_i^* 's are broadcasted the simulator does not have to "match" the public commitments generated by the Joint-RVSS.
- 5. Values $R_i, R'_i, i \in \mathcal{G}$ satisfy the required constraints (i.e. verification equation) as in the protocol.

We have shown that the *public* view of the adversary during the simulation is identical to the one he would see during a real execution. Now we must proceed to show that the simulator can produce a consistent view of the internal states for the players corrupted by the adversary \mathcal{A} . Clearly, if a player is corrupted before Step 2, the simulator can produce a consistent view because it is following the protocol. After Step 2 the simulator can show correct internal states for all the players in \mathcal{G} except for the special player P. Thus, if P is corrupted the simulator rewinds the adversary to the beginning of Step 2 and selects at random a different special player. Notice that if a player P_i in $\mathcal{G} \setminus \{P\}$ is corrupted after Step 4, the simulator has broadcasted for P_i a "fake" value d_i^* . But since we erased all the private information (except the shares) generated in Joint-RVSS in Step 3, the simulator can simply claim that d_i^* was really the share held by P_i . This will not contradict any of the generated public information.

Adaptive vs. Non-Adaptive Solutions. As noted before, DKG is based on the recent distributed key generation protocol of [GJKR99], which is secure only against a static adversary. The generation of x is the same in both protocols but

they differ in the method for extracting the public key. In the current protocol each player reveals only values $(A_i, B_i) = (g^{f_i(0)}, h^{f_i'(0)})$ from the execution of Joint-RVSS, and uses zero-knowledge proofs to guarantee that these values are properly formed. Due to this limited revealing of information it is sufficient for the simulator to "cheat" with respect to the internal state of only a single player, and yet to "hit" the desired value. However, in the protocol of [GJKR99], each player reveals A_i by publishing all values $g^{a_{ik}}$, k=0,...,t. For one of the players the simulator has to commit in this way to a polynomial without knowing $a_{i0} = f_i(0)$. Therefore he can do it in a way that is consistent with only t points on this polynomial. Thus, the simulator has an inconsistent internal state for n-t players, and hence has to stop and rewind every time one of them is corrupted, which happens with overwhelming probability if the adversary is adaptive.

Key Generation with Optimal Resilience. To achieve an optimally-resilient (i.e. n=2t+1) DKG protocol two changes are required. First we need to change the generation of h which occurs before DKG starts. Instead of using a VSS protocol which has a t < n/3 resilience ([BGW88]) we need a VSS with an optimal t < n/2 resilience (e.g. [CDD⁺99]). The second change is the following: In our DKG we publicly reconstruct a value created with Joint-RVSS protocol (see Steps 3-4). This reconstruction is currently done using error-correcting codes, which make the protocol easy to simulate, but which tolerate only t < n/3 faults. However, we can achieve optimal resilience by sieving out bad shares with Pedersen verification equation (Eq. (1)) if the players submit the associated random values generated by Joint-RVSS together with their shares. Therefore the players must no longer erase these values as in the current Step 3.

This change must be reflected in the simulator, because the current SIM-DKG is unable to produce these values. However, the simulator could produce them if he knew the discrete logarithm $Dlog_g(h)$. Therefore, in the h-generation protocol, instead of playing the part of the honest parties, the simulator must pick $\lambda \in \mathbb{Z}_q$ at random, compute $h = g^{\lambda}$, simulate the VSS protocol of [CDD⁺99] to arrive at $r = h^{\beta}$ where $\beta = ((p-1)/q)^{-1} \mod q$, and pass λ to the modified SIM-DKG.

4 Adaptively-Secure Threshold DSS Signature Scheme

As described in Section 2, a Threshold Signature Scheme consists of a distributed key generation, distributed signature protocol, and a signature verification procedure (see full definitions in [GJKR96] or [CGJ⁺]). Here we present a distributed DSS signature protocol Sig-Gen with t < n/4 resilience. (Below we give a brief description for how to modify this protocol to achieve optimal resilience.) We prove the unforgeability of the Threshold DSS Signature Scheme, which is combined from DKG (Section 3), Sig-Gen, and the regular DSS verification procedure DSS-Ver. The proof of robustness is deferred to [CGJ⁺]. We refer the reader to Section 2 for a higher-level description of the basic elements in our approach, solutions, and proofs.

Distributed Signature Protocol. The basis for the signature protocol Sig-Gen (Fig.4) is the protocol of [GJKR96], with modifications to allow for the adaptive

adversary. Protocol Sig-Gen assumes that the signing parties have previously executed the DKG protocol and hold the shares x_i of a secret key that corresponds to the generated public key y. The protocol Sig-Gen is invoked every time some message m needs to be signed.

Input: message m to be signed (plus the outputs of DKG). Public Output: (r, s) the signature of the message m

- 1. Generate $r = g^{k^{-1}} \mod p \mod q$
 - (a) Generate k. Players execute Joint-RVSS(t, n, t). Player P_i sets k_i to his share of the secret. All other secret information generated by the above execution is erased.
 - (b) Generate random sharings of 0 on polynomials of degree 2t. Players execute two instances of Joint-ZVSS(t, n, 2t). Player P_i sets b_i and c_i to his shares of the two sharings. All other secret information generated by the above executions is erased.
 - (c) Generate a random value a and $g^a \mod p$ using DKG. Player P_i sets a_i to his share of the secret a.
 - (d) Player P_i broadcasts $v_i = k_i a_i + b_i \mod q$.
 - (e) Each player computes: $\mu \stackrel{\triangle}{=} \mathsf{EC-Interpolate}(v_1,\ldots,v_n) \bmod q \ [= ka \bmod q]$, then $\mu^{-1} \bmod q$, and $r \stackrel{\triangle}{=} (g^a)^{\mu^{-1}} [= g^{k^{-1}}] \bmod p \bmod q$.
- 2. Generate $s = k(m + xr) \mod q$
 - P_i broadcasts $s_i = k_i(m + x_i r) + c_i \mod q$. Set $s \stackrel{\triangle}{=} \text{EC-Interpolate}(s_1, \ldots, s_n)$.
- 3. Player P_i erases all secret information generated in this protocol.

Fig. 4. Sig-Gen - Distributed Signature Generation, $n \ge 4t + 1$

The first part of the signature computation is the generation of the random value $r=g^{k^{-1}} \mod p \mod q$. This computation is very similar to the distributed key generation protocol, aside from the complication that it requires g to be raised to the inverse of the shared secret value k. We achieve this with a variation of the distributed inversion protocol from [BB89]: The players select a random uniformly distributed $k \in Z_q$ in shared form via the Joint-RVSS protocol (Step 1a). Then they perform a DKG protocol to select a random $a \in Z_q$ in shared form and publish $g^a \mod p$ (Step 1c). The inversion of k in the exponent occurs when the players reconstruct in the clear the product $\mu = ka \mod q$ which is a random number, invert it, and then compute $g^{k^{-1}} = (g^a)^{\mu^{-1}}$ (Steps 1d-1e). The value s is publicly reconstructed when each player reveals the product $k_i(m+x_ir)$ which lies on a 2t-degree polynomial whose free term is s = k(m+xr). As in DKG, it is crucial for the proof of adaptive security that the players erase at the end all secret information generated by this protocol.

Sig-Gen uses a Joint-ZVSS subprotocol to generate randomizing polynomials (Step 1b). This randomization is needed to hide all partial information during the public reconstruction of values μ and s (Steps 1e and 2). Joint-ZVSS, which stands for "Joint Zero VSS", is a modification of Joint-RVSS where all players

fix their values $z_i = a_{i0}$ and b_{i0} (Step 1a, Figure 2) to zero. This can be verified by other players by checking that $C_{i0} = 1 \mod p$.

Proving Adaptive Security of Threshold DSS. We argue the following:

Theorem 1. If DSS is unforgeable under adaptive chosen message attack then DSS-ts=(DKG,Sig-Gen,DSS-Ver) is a secure (unforgeable and robust) (t,n) -threshold signature scheme for t < n/4.

Due to space limitations we present only the unforgeability part of this argument (Lemma 3) and we use the definition of unforgeability of threshold signature schemes from [GJKR96]. For the complete treatment of security of our scheme, together with more formal definitions of security of threshold signatures and a proof of robustness, we invite the reader to [CGJ⁺]. To prove unforgeability, we first need the following lemma about the simulator SIM-Sig presented in Fig.5:

Input: message m, its signature (r, s) of a DSS system (y, p, q, g), an element h generated by g

Compute $r^* = g^{ms^{-1}}y^{rs^{-1}} \mod p$. Pick $\mu \in_R Z_q$, and set $\beta = (r^*)^{\mu} \mod p$.

- 1. (a) Execute Joint-RVSS(t, n, t) on behalf of the uncorrupted players. This results in shares k_i for each of the uncorrupted players. Erase all secret values aside from k_i .
 - (b) Execute two sharings of a 0 value on polynomials of degree 2t using Joint-ZVSS(t, n, 2t). This results in shares b_i, c_i for the uncorrupted players. Erase all secret values aside from b_i, c_i .
 - (c) Run the simulator SIM-DKG of the DKG protocol on input β as the "public key". This results in shares a_i for each of the uncorrupted players. Note that all other information has already been erased by the simulator which was called as a sub-routine.
 - (d) The simulator knows values $v_i = k_i a_i + b_i$, $i \in \mathcal{B}$ that should be broadcast by the players controlled by the adversary in Step 1e of the signing protocol. Choose a 2t-degree polynomial $f_v(z)$ s.t. $f_v(0) = \mu$, $f_v(i) = v_i$ for $i \in \mathcal{B}$. Set $v_i^* = f_v(i)$ for $i \in \mathcal{G}$. Compute $b_i^* = v_i^* k_i a_i$, $i \in \mathcal{G}$. Erase the secret values b_i for $i \in \mathcal{G}$. Broadcast v_i^* for $i \in \mathcal{G}$.
- 2. The simulator knows values $s_i = k_i(m+x_ir) + c_i$, $i \in \mathcal{B}$ that should be broadcast by the players controlled by the adversary in Step 2 of the signing protocol. Choose a 2t-degree polynomial $f_s(z)$ s.t. $f_s(0) = s$, $f_s(i) = s_i$ for $i \in \mathcal{B}$. Set $s_i^* = f_s(i)$ for $i \in \mathcal{G}$. Compute $c_i^* = s_i^* k_i(m+x_ir)$, $i \in \mathcal{G}$. Erase the secret values c_i for $i \in \mathcal{G}$. Broadcast s_i^* for $i \in \mathcal{G}$.
- 3. Erase all the information generated by the signature generation.

 $\mathbf{Fig.5.}$ SIM-Sig - Simulator for the Distributed Signature Protocol Sig-Gen

Lemma 2. Simulator SIM-Sig on input (p, q, g, h, y, m, (r, s)) ends in expected polynomial time and computes a view for the adversary that is indistinguishable from a view of the protocol Sig-Gen on input public key (p, q, g, h, y), message m, and output signature (r, s).

Proof: 1. Generation of r

- (a) The simulator executed Joint-RVSS according to the protocol, thus all the values generated by these executions have the required distribution.
- (b) The simulator executed two instances of Joint-ZVSS according to the protocol, thus all the values generated by these executions have the required distribution.
- (c) It has been proved that the simulator SIM-DKG outputs the desired distributions for the DKG protocol. Notice also that β is uniformly distributed in G_g (the same as g^a in the real protocol).
- (d) Here the simulator broadcasts values v_i^* which were not computed according to the protocol. Yet because these shares have been chosen at random under the condition that they interpolate to a random free term μ and the polynomial interpolated by them matches the shares held by the adversary, the view of the adversary is exactly the same as in the real protocol.
- 2. Generation of s: The same argument as above applies to the values s_i^* .

The discussion about the internal states presented by the simulator to the adversary when a player is corrupted is identical to the one in the proof of Lemma 1. It is important to notice that the only rewinding happens during the simulation of the DKG subroutine inside this protocol. It should be restated here that once the DKG simulator completes its execution the adversary can now corrupt any of the players, including the "special" player, because now even for that player the simulator has a consistent view.

Lemma 3. If DSS is unforgeable under adaptive chosen message attack then DSS-ts=(DKG, Sig-Gen, DSS-Ver) is an unforgeable (t, n)-threshold signature scheme for t < n/4.

Proof: Assume that DSS-ts is not unforgeable. Then there exists a t-threshold adversary \mathcal{A} s.t. with a non-negligible probability \mathcal{A} outputs a valid DSS (message, signature) pair (m, (r, s)), after participating in the initial execution of DKG and then in the repeated execution of Sig-Gen on messages $m_1, m_2, ...$ of \mathcal{A} 's choice. Furthermore, none of the m_i 's is equal to m. Using such adversary \mathcal{A} , we show how to construct a forger \mathcal{F} against the regular DSS scheme. \mathcal{F} is given as input a DSS system (p, q, g), and a random public key y. Additionally, \mathcal{F} can accesses a signature oracle O_{Sig} that provides DSS signatures under the given public key (p, q, g, y).

 \mathcal{F} fixes the random coins of \mathcal{A} . First \mathcal{F} plays the part of the honest parties in the h-generation protocol. Then \mathcal{F} runs an interaction between SIM-DKG and \mathcal{A} on input (p,q,g,h,y). By lemma 1, the simulation ends in expected polynomial time and \mathcal{A} receives a view that is identical to \mathcal{A} 's view of a random execution of DKG that outputs y. When \mathcal{A} requests a signature on message m_i , \mathcal{F} submits m_i to O_{Sig} and receives (r_i,s_i) , a random DSS signature on m_i . Then \mathcal{F} then runs an interaction between SIM-Sig and \mathcal{A} on input $(p,q,g,h,y,m_i,(r_i,s_i))$. By lemma 2, this simulation ends in expected polynomial time and \mathcal{A} receives a view that is identical to \mathcal{A} 's view of a random execution of Sig-Gen that outputs (r_i,s_i) . Finally, since its views of this simulation are indistinguishable from the

real ones, \mathcal{A} outputs a valid DSS signature (m,(r,s)) where $m \neq m_i$ for all i. \mathcal{F} outputs this (m,(r,s)) which is a successful existential forgery since \mathcal{F} never asked its oracle this message m.

Threshold DSS with Optimal t < n/2 Resilience. Because of lack of space we defer the description of the modifications that achieve optimal resilience in the Sig-Gen protocol to the full version [CGJ⁺] of this paper. These modifications include the use of the t < n/2 resilient version of DKG as described at the end of section 3. Furthermore, we use the results of [GRR98] to increase the resilience of the secret-multiplication steps of the signature protocol itself.

5 Further Applications

The techniques introduced in this paper enable us to achieve adaptive security for other threshold public-key systems. Here we sketch our solution to Adaptive Threshold RSA and Adaptive Proactive Solutions.

Adaptive Threshold RSA. We can achieve an adaptively-secure Threshold RSA Signature Generation protocol (but without distributed key generation) with optimal resilience. Furthermore, our protocol runs in a constant number of rounds. We build our solution on the Threshold RSA solution of [Rab98]. The protocol of that paper needs to be modified to use a Pedersen VSS wherever a Feldman VSS is used, the zero-knowledge proofs which appear need to be modified accordingly, and commitments should be of the Pedersen form.

The most interesting change required in this protocol is due to the following: If the current protocol is invoked twice on two different messages, each player gives its partial signature on these messages under its fixed partial key. But as we have seen in Section 2, the simulator is not allowed to rewind back beyond the current invocation of the signature protocol that it is simulating. Clearly, the simulator cannot know the partial keys of all players. If a player for whom he does not know the partial key is broken into during the simulation of the signing of the second message, then the simulator would need to explain both partial signatures of that player, and hence would be forced to rewind beyond the current invocation of the signature protocol.

To avoid this problem, the partial keys need to be changed after each signature generation. This is achieved in a straightforward manner (though it adds a performance penalty to the protocol).

Adaptive Proactive Solutions. Our threshold DSS scheme, as well as other discrete—log based threshold schemes built with our techniques, can be easily proactivized $[HJJ^+97]$ by periodic refreshment of the shared secret key. Due to space limitations we simply state here that such refreshment can be achieved in the adaptive model if the players execute a Joint-ZVSS protocol (see Section 4), and add the generated shares to their current share of the private key x.

References

- BB89. J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds. In *Proc.* 8th ACM PODC, pages 201–209. ACM, 1989.
- BGW88. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations. In *Proc.* 20th STOC, pages 1–10. ACM, 1988.
- BH92. D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Eurocrypt '92*, pages 307–323, 1992. LNCS No. 658.
- BW. E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent 4,633,470.
- Can98. R. Canetti. Security and composition of multiparty cryptographic protocols. Available at the Theory of Cryptography Library, http://theory.lcs.mit.edu/~tcryptol. 1998.
- CCD88. D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. In Proc. 20th STOC, pages 11–19. ACM, 1988.
- CD98. R. Cramer and I. Damgard. Zero-knowledge proof for finite fields arithmetic, or: Can zero-knowledge be for free. In Crypto '98, pages 424–441, 1998. LNCS No. 1462.
- CDD⁺99. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations with dishonest minority. In *Eurocrypt '99*, pages 311–326, 1999. LNCS No.
- CFGN96. Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proc.* 28th STOC, pages 639–648. ACM, 1996.
- CGJ^{+} . S. Jarecki, Krawczyk, R. Canetti, R. Gennaro, Η. Τ. Rabin. Adaptive security for threshold cryptosystems. http://www.research.ibm.com/security/adss.ps.
- DNS98. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *Proc.* 30th STOC, pages 409–418. ACM, 1998.
- FMY. Y. Frankel, P. MacKenzie, and M. Yung. Adaptively-secure distributed public-key systems. Personal communication with M.Yung.
- GJKR96. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Eurocrypt '96*, pages 354–371, 1996. Full version: http://www.research.ibm.com/security/DSSthresh.ps.
- GJKR99. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. The (in)security of distributed key generation in dlog-based cryptosystems. In *Eurocrypt '99*, pages 295–310, 1999. LNCS No.
- GK96. O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. SIAM. J. Computing, 25(1), 1996.
- GMR88. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Computing, 17(2):281–308, April 1988.
- GMR89. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. SIAM. J. Computing, 18(1):186–208, February 1989.
- GRR98. R. Gennaro, M. Rabin, and T Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In Proc. 17th ACM PODC, pages 101–112. ACM, 1998.
- HJJ⁺97. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In 1997 ACM Conference on Computers and Communication Security, 1997.

- IS93. T. Itoh and K. Sakurai. On the complexity of constant round zkip of possession of knowledge. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E76-A(1), January 1993.
- KPR98. J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero-knowledge on the internet. In *Proc.* 39th FOCS, pages 484–492. IEEE, 1998.
- Ped91a. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Crypto '91, pages 129–140, 1991. LNCS No. 576.
- Ped91b. T. Pedersen. A threshold cryptosystem without a trusted party. In *Eurocrypt '91*, pages 522–526, 1991. LNCS No. 547.
- Rab98. Tal Rabin. A simplified approach to threshold and proactive RSA. In *Crypto* '98, pages 89–104, 1998. LNCS No. 1462.
- Sch91. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

Two Party RSA Key Generation

(Extended Abstract)

Niv Gilboa

Computer Science Department Technion, Haifa 32000, Israel gilboa@cs.technion.ac.il

Abstract. We present a protocol for two parties to generate an RSA key in a distributed manner. At the end of the protocol the public key: a modulus N=PQ, and an encryption exponent e are known to both parties. Individually, neither party obtains information about the decryption key d and the prime factors of N: P and Q. However, d is shared among the parties so that threshold decryption is possible.

1 Introduction

We show how two parties can jointly generate RSA public and private keys. Following the execution of our protocol each party learns the public key: N = PQ and e, but does not know the factorization of N or the decryption exponent d. The exponent d is shared among the two players in such a way that joint decryption of cipher-texts is possible.

Generation of RSA keys in a private, distributed manner figures prominently in several cryptographic protocols. An example is threshold cryptography, see [12] for a survey. In a threshold RSA signature scheme there are k parties who share the RSA keys in such a way that any t of them can sign a message, but no subset of at most t-1 players can generate a signature. A solution to this problem is presented in [11]. An important requirement in that work is that both the public modulus N and the private key are generated by a dealer and subsequently distributed to the parties. The weakness of this model is that there is a single point of failure—the dealer himself. Any adversary who compromises the dealer can learn all the necessary information and in particular forge signatures.

Boneh and Franklin show in [4] how to generate the keys without a dealer's help. Therefore, an adversary has to subvert a large enough coalition of the participants in order to forge signatures. Several specific phases of the Boneh-Franklin protocol utilize reduced and optimized versions of information theoretically private multi-party computations [1,6]. Those phases require at least three participants: Alice and Bob who share the secret key and Henry, a helper party, who knows at the end of the protocol only the public RSA modulus N.

Subsequent works [8,9,20] and [13] consider other variants of the problem of jointly generating RSA keys. In [8] Cocks proposes a method for two parties

to jointly generate a key. He extends his technique to an arbitrary number of parties k [9]. The proposed protocol suffers from several drawbacks. The first is that the security is unproven and as Coppersmith pointed out (see [9]) the privacy of the players may be compromised in certain situations. The second is that the protocol is far less efficient than the Boneh-Franklin protocol. In [20] Poupard and Stern show a different technique for two parties to jointly generate a key. Their method has proven security given standard cryptographic assumptions. Some of the techniques employed in the current work are similar to the ideas of [20] but the emphasis is different. Poupard and Stern focus on maintaining robustness of the protocol, while we emphasize efficiency. In [13] Frankel, Mackenzie and Yung investigate a model of malicious adversaries as opposed to the passive adversaries considered in [4,8,9] and in our work. They show how to jointly generate the keys in the presence of any minority of misbehaving parties.

The current work focuses on joint generation of RSA keys by two parties. We use the Boneh-Franklin protocol and replace each three party sub-protocol with a two party sub-protocol. We construct three protocols. The first is based on $\binom{2}{1}$ oblivious transfer of strings. Thus, its security guarantee is similar to that of general circuit evaluation techniques [22,16]. The protocol is more efficient than the general techniques and is approximately on par with Cocks' method and slightly faster than the Poupard-Stern method. The second utilizes a new intractability assumption akin to noisy polynomial reconstruction that was proposed in [19]. The third protocol is based on a certain type of homomorphic encryption function (a concrete example is given By Benaloh in [2,3]). This protocol is significantly more efficient than the others both in computation and communication. It's running time is (by a rough estimate) about 10 times the running time the Boneh-Franklin protocol.

There are several reasons for using 3 different protocols and assumptions. The first assumption is the mildest one may hope to use. The second protocol has the appealing property that unlike the other two protocols it is not affected by the size of the moduli. In other words the larger the RSA modulus being used, the more efficient this protocol becomes in comparison with the others. Another interesting property of the first two protocols is that a good solution to an open problem we state at the end of the paper may make them more efficient in terms of computation than the homomorphic encryption protocol.

We assume that an adversary is passive and static. In other words, the two parties follow the protocol to the letter. An adversary who compromises a party may only try to learn extra information about the other party through its view of the communication. Furthermore, an adversary who takes over Alice at some point in the execution of the protocol cannot switch over to Bob later on, and vice versa.

The remainder of the work is organized as follows. In section 2 we describe some of the tools and techniques developed previously and used in this paper. In section 3 we give an overview of the Boneh-Franklin protocol and of where we diverge from it. In section 4 we describe how to compute a modulus N in a distributed fashion using each of the three protocols. In section 5 we show

how to amortize the cost of computing several candidate moduli N and how to perform trial divisions of small primes. In section 6 we show how to compute the decryption exponent d. In section 7 we describe how to improve the efficiency of the homomorphic encryption protocol. In section 8 we discuss some performance issues.

2 Preliminaries

Notation 1: The size of the RSA modulus N is σ bits (e.g $\sigma = 1024$).

2.1 Smallest Primes

At several points in our work we are interested in the j smallest distinct primes p_1, \ldots, p_j such that $\pi_{i=1}^j p_i > 2^{\sigma}$. The following table provides several useful parameters for a few typical values of σ .

	σ	j	p_{j}	$\sum_{i=1}^{j} \lceil \log p_i \rceil$
I	512	76	383	557
	1024	133	751	1108
	1536	185	1103	1634
	2048	235	1483	2189

2.2 Useful Techniques

In this subsection we review several problems and techniques that were researched extensively in previous work, and which we use here.

Symmetrically private information retrieval: In the problem of private information retrieval presented by Chor et al. [7] k databases $(k \geq 1)$ hold copies of the same n bit binary string x and a user wishes to retrieve the i-th bit x_i . A PIR scheme is a protocol which allows the user to learn x_i without revealing any information about i to any individual database. Symmetrically private information retrieval, introduced in [15], is identical to PIR except for the additional requirement that the user learn no information about x apart from x_i . This problem is also called 1 out of m oblivious transfer and all or nothing disclosure of secrets (ANDOS). The techniques presented in [15] are especially suited for the multi-database $(k \geq 2)$ setting. In a recent work Naor and Pinkas [19] solve this problem by constructing a SPIR scheme out of any PIR scheme, in particular single database PIR schemes. The trivial single database PIR scheme is to simply have the database send the whole data string to the user. Clever PIR schemes involving a single database have been proposed in several works: [18,5,21]. They rely on a variety of cryptographic assumptions and share the property that for "small" values of n their communication complexity is worse than that of the trivial PIR scheme.

We now give a brief description of the SPIR scheme we use, which is the Naor-Pinkas method in conjunction with the trivial PIR scheme. In our scenario the data string x is made up of n substrings of length ℓ . The user retrieves the i-th substring without learning any other information and without leaking information about i. The database randomly chooses $\log n$ pairs of seeds for a pseudo-random generator $G\colon (s_1^0,s_1^1),\ldots,(s_{\log n}^0,s_{\log n}^1,s_{\log n}^1)$. Every seed s_j^b $(1\leq j\leq \log n,\ b\in\{0,1\})$ is expanded into $n\ell$ bits $G(s_j^b)$, which can be viewed as n substrings of length ℓ . It then prepares a new data string y of n substrings. Suppose the binary representation of i is $i_{\log n}\ldots i_1$. The i-th substring of y is the exclusive-or of the i-th substring of x and the i-th substring of each of $G(s_1^{i_1}), G(s_2^{i_2}), \ldots, G(s_{\log n}^{i_{\log n}})$. The user and database combine in $\log n$ $\binom{2}{1}$ -OT of strings to provide the user with a single seed from every pair. Finally, the database sends y to the user, who is now able to learn a single substring of x. The parameters of the data strings we use are such that the running time is dominated by the $\log n$ $\binom{2}{1}$ -OTs and the communication complexity is dominated by the $n\ell$ bits of the data string, which are sent to the user.

Dense probabilistic and homomorphic encryption: we are interested in an encryption method that provides two basic properties: (1) Semantic security: as defined in [17]. (2) Additive homomorphism: we can efficiently compute a function f such that f(ENC(a), ENC(b)) = ENC(a+b). Furthermore, the sum is modulo some number t, where t can be defined flexibly as part of the system.

As a concrete example we use Benaloh's encryption [2,3]. The system works as follows. Select two primes p,q such that: $m \stackrel{\triangle}{=} pq \approx 2^{\sigma}$, t|p-1, $\gcd(t,(p-1)/t) = 1$ and $\gcd(t,q-1) = 1^{-1}$. The density of such primes along appropriate arithmetic sequences is large enough to ensure efficient generation of p,q (see [2] for details). Select $y \in \mathcal{Z}_m^*$ such that $y^{\phi(m)/t} \not\equiv 1 \mod m$. The public key is m, y, and encryption of $M \in \mathcal{Z}_t$ is performed by choosing a random $u \in \mathcal{Z}_m^*$ and sending $y^M u^t \mod m$.

In order to decrypt, the holder of the secret key computes at a preprocessing stage $T_M \stackrel{\triangle}{=} y^{M\phi(m)/t} \mod m$ for every $M \in \mathcal{Z}_t$. Hence, t is small enough that t exponentiations can be performed. Decryption of z is by computing $z^{\phi(m)/t} \mod m$ and finding the unique T_M to which it is equal. The scheme is semantically secure based on the assumption that deciding higher residuosity is intractable [3]. Most of our requirements are met by the weaker assumption that deciding prime residuosity is intractable [2].

Oblivious polynomial evaluation: In this problem, presented by Naor and Pinkas in [19] Alice holds a field element $\alpha \in \mathcal{F}$ and Bob holds a polynomial B(x) over \mathcal{F} . At the end of the protocol Alice learns only $B(\alpha)$ and Bob learns nothing at all. The intractability assumption used in [19] is new and states the following. Let $S(\cdot)$ is a degree k polynomial over \mathcal{F} and let $m, d_{Q,x}$ be two security parameters $(d_{Q,x} > k)$. Given $2d_{Q,x} + 1$ sets of m field elements such that in each set there is one value of S at a unique point different than 0 and m-1 random field elements, the value S(0) is pseudo-random.

We now give a brief description of the protocol presented in [19] as used in our application where the polynomial B is of degree 1. Bob chooses a random

¹ Therefore t is odd.

bivariate polynomial Q(x,y) such that the degree of y is 1, the degree of x is $d_{Q,x}$ and $Q(0,\cdot)=B(\cdot)$. Alice chooses a random polynomial S of degree $d_{Q,x}$ such that $S(0)=\alpha$. Define R(x) as the degree $2d_{Q,x}$ polynomial R(x)=Q(x,S(x)). Alice chooses $2d_{Q,x}+1$ different non-zero points x_j for $j=1,\ldots,2d_{Q,x}+1$. For each such j Alice randomly selects m-1 field elements $y_{j,1},\ldots,y_{j,m-1}$ and sends to Bob x_j and a random permutation of the m elements $S(x_j),y_{j,1},\ldots,y_{j,m-1}$ (denoted by $z_{j,1},\ldots,z_{j,m}$). Bob computes $Q(x_j,z_{j,i})$ for $i=1,\ldots,m$. Alice and Bob execute a SPIR scheme in which Alice retrieves $Q(x_j,S(x_j))$. Given $2d_{Q,x}+1$ such pairs of $x_j,R(x_j)$ Alice can interpolate and compute $R(0)=B(\alpha)$.

The complexity of the protocol is $2d_{Q,x} + 1$ executions of the SPIR scheme for data strings of m elements.

3 Overview

In this section we give an overview of our protocol. The stages in which we use the Boneh-Franklin protocol exactly are the selection of candidates and the full primality test (the other stages require a third party in [4]). The protocol is executed in the following steps.

- 1. Choosing candidates Alice chooses independently at random two $\sigma/2-1$ bit integers $P_a, Q_a \equiv 3 \mod 4$, and Bob chooses similarly $P_b, Q_b \equiv 0 \mod 4$. The two parties keep their choices secret and set as candidates $P = P_a + P_b$ and $Q = Q_a + Q_b$.
- 2. Computing N Alice and Bob compute $N = (P_a + P_b)(Q_a + Q_b)$. We show how to perform the computation using three different protocols and three different intractability assumptions.
- 3. **Initial primality test** For each of the smallest k primes p_1, \ldots, p_k the participants check if $p_i \mid N$ $(i = 1, \ldots, k)$. This stage is executed in conjunction with the computation of N. If N fails the initial primality test, computing a new candidate N is easier than computing it from scratch (as is the case following a failure of the full primality test)
- 4. Full primality test The test of [4] is essentially as follows: Alice and Bob agree on $g \in \mathcal{Z}_N^*$. If the Jacobi symbol $\left(\frac{g}{N}\right)$ is not equal to 1 choose a new g. Otherwise Alice computes $v_a = g^{(N-P_a-Q_a+1)/4} \mod N$, and Bob computes $v_b = g^{(P_b+Q_b)/4} \mod N$. If $v_a = v_b$ or $v_a = -v_b \mod N$ the test passes.
- 5. Computing and sharing d In this step we compute the decryption exponent d assuming that e is known to both parties and that $gcd(e, \phi(N)) = 1$. Alice receives d_a and Bob receives d_b so that $d = d_a + d_b \mod \phi(N)$ and $de \equiv 1 \mod m$. Boneh and Franklin describe two protocols for the computation of d. The first is very efficient and can be performed by two parties, but leaks $\phi(n) \mod e$. Therefore, this method is suitable for small public exponents and not for the general case. The second protocol computes d for any e but requires the help of a third party.

4 Computing N

Alice holds P_a , Q_a and Bob holds P_b , Q_b . They wish to compute

$$N = (P_a + P_b)(Q_a + Q_b) = P_aQ_a + P_aQ_b + P_bQ_a + P_bQ_b.$$

We show how to carry out the computation privately using three different protocols.

4.1 Oblivious Transfers

Let \mathcal{R} be a publicly known ring and let $a, b \in \mathcal{R}$. Denote $\rho = \log |\mathcal{R}|$ (each element in \mathcal{R} can be encoded using ρ bits). Assume Alice holds a and Bob holds b. They wish to perform a computation by which Alice obtains x and Bob obtains y such that x + y = ab where all operations are in \mathcal{R} . Furthermore, the protocol ensures the privacy of each player given the existence of oblivious transfers. In other words the protocol does not help Alice and Bob to obtain information about b and a respectively. The protocol:

- 1. Bob selects uniformly at random and independently ρ ring elements denoted by $s_0, \ldots, s_{\rho-1} \in \mathcal{R}$. Bob proceeds by preparing ρ pairs of elements in \mathcal{R} : $(t_0^0, t_0^1), \ldots, (t_{\rho-1}^0, t_{\rho-1}^1)$. For every i $(0 \le i \le \rho 1)$ Bob defines $t_i^0 \stackrel{\triangle}{=} s_i$ and $t_i^1 = 2^i b + s_i$.
- 2. Let the binary representation of a be $a_{\rho-1} \dots a_0$. Alice and Bob execute ρ $\binom{2}{1}$ -OTs. In the i-th invocation Alice chooses $t_i^{a_i}$ from the pair (t_i^0, t_i^1) .
- 3. Alice sets $x \stackrel{\triangle}{=} \sum_{i=0}^{\rho-1} t_i^{a_i}$ and Bob sets $y \stackrel{\triangle}{=} -\sum_{i=0}^{\rho-1} s_i$.

Lemma 1. x + y = ab over the ring \mathcal{R} .

Proof. Since $a_{\rho-1}, \ldots, a_0$ is the binary representation of a we can write $a = \sum_{i=0}^{\rho-1} a_i \cdot 2^i$.

$$x + y = \sum_{i=0}^{\rho-1} t_i^{a_i} - \sum_{i=0}^{\rho-1} s_i$$

$$\equiv \sum_{i=0}^{\rho-1} (a_i \cdot 2^i b + s_i) - \sum_{i=0}^{\rho-1} s_i$$

$$\equiv b \sum_{i=0}^{\rho-1} a_i \cdot 2^i$$

$$= ab$$

In the following protocol for computing N the ring \mathcal{R} is $\mathcal{Z}_{2^{\sigma}}$, the integers modulo 2^{σ} .

- 1. Alice and Bob use the previous protocol twice to additively share $P_aQ_b=x_1+y_1 \bmod 2^{\sigma}$ and $P_bQ_a=x_2+y_2 \bmod 2^{\sigma}$. Alice holds x_1,x_2 and Bob holds y_1,y_2 .
- 2. Bob sends $y \stackrel{\triangle}{=} y_1 + y_2 + P_b Q_b \mod 2^{\sigma}$ to Alice.
- 3. Alice computes $P_aQ_a+y \mod 2^{\sigma}$. Alice now holds $N \mod 2^{\sigma}$, which is simply N due to the choice of σ .

Lemma 2. The transcript of the view of the execution of the protocol can be simulated for both Alice and Bob and therefore the protocol is secure.

Proof. We denote the messages that Alice receives during the sharing of P_aQ_b by $t_0^{a_0}, \ldots, t_{\sigma-1}^{a_{\sigma-1}}$ and the messages received while sharing P_bQ_a by $t_{\sigma}^{a_{\sigma}}, \ldots, t_{2\sigma-1}^{a_{2\sigma-1}}$. In the same manner we denote Bob's random choices for the sharing of P_aQ_b and P_bQ_a by $s_0, \ldots, s_{\sigma-1}$ and $s_{\sigma}, \ldots, s_{2\sigma-1}$ respectively.

Bob's view can be simulated because the only messages Alice sent him were her part of 2σ independent oblivious transfers.

Alice receives $2\sigma + 1$ elements in $\mathcal{Z}_{2\sigma}$:

$$t_0^{a_0}, \ldots, t_{2\sigma-1}^{a_{2\sigma-1}}, y.$$

The uniformly random and independent choices by which $s_0, \ldots, s_{2\sigma-1}$ are selected ensure that the messages Alice receives are distributed uniformly subject to the condition that

$$\sum_{i=0}^{2\sigma-1} t_i^{a_i} + y \equiv N - P_a Q_a \bmod 2^{\sigma}.$$

Since Alice can compute $N - P_a Q_a$ a simulator S_a can produce the same distribution as that of the messages Alice receives, given N, P_a, Q_a .

Lemma 3. The computation time and the communication complexity of the protocol are dominated by 2σ oblivious transfers. The transfered strings are of length σ .

4.2 Oblivious Polynomial Evaluation

Alice and Bob agree on a prime $p > 2^{\sigma}$ and set \mathcal{F} to be GF(p). They employ the following protocol to compute N:

- 1. Bob chooses a random element $r \in \mathcal{F}$. He prepares two polynomials over \mathcal{F} : $B_1(x) = P_b x + r$ and $B_2 = Q_b x r + P_b Q_b$.
- 2. Alice uses the oblivious polynomial evaluation protocol of [19] to attain $B_1(Q_a)$ and $B_2(P_a)$. Alice computes $N = P_a Q_a + B_1(Q_a) + B_2(P_a)$.

The security of the protocol depends on the security of the cryptographic assumption outlined in subsection 2.2 and of a similar argument to the proof of lemma 2.

Lemma 4. The computational complexity of the protocol is dominated by the execution of $2 \log m(2d_{Q,x} + 1)$ oblivious transfers, where m and $d_{Q,x}$ are the security parameters. The communication complexity is less than $3m(2d_{Q,x} + 1)\sigma$.

4.3 Benaloh's Encryption

We now compute N by using the homomorphic encryption described in subsection 2.2. Let p_1, \ldots, p_j be the smallest primes such that $\pi_{i=1}^j p_i > 2^{\sigma}$. The following protocol is used to compute $N \mod p_i$:

- 1. Let $t \stackrel{\triangle}{=} p_i$. Alice constructs the encryption system: an appropriate p, q, y, and sends the public key y, m = pq to Bob. Alice also sends the encryption of her shares, i.e $z_1 \stackrel{\triangle}{=} y^{P_a} u_1^t \mod m$ and $z_2 \stackrel{\triangle}{=} y^{Q_a} u_2^t \mod m$, where $u_1, u_2 \in \mathcal{Z}_m^*$ are selected uniformly at random and independently.
- 2. Bob computes the encryption of $P_bQ_b \mod t$, which is denoted by z_3 , calculates

$$z \stackrel{\triangle}{=} z_1^{Q_b} \cdot z_2^{P_b} \cdot z_3 \bmod m$$

and sends z to Alice.

3. Alice decrypts z, adds to the result P_aQ_a modulo t and obtains N mod t.

The two players repeat this protocol for each p_i , i = 1,...,j. Alice is able to reconstruct N from $N \mod p_i$, i = 1,...,j by using the Chinese remainder theorem.

Lemma 5. Assuming the intractability of the prime residuosity problem, the transcript of the views of both parties in the protocol can be simulated.

Proof. The distribution of Bob's view can be simulated by encrypting two arbitrary messages assuming the intractability of prime residuosity. Therefore, Alice's privacy is assured.

The distribution of Alice's view can be simulated as follows. Given $N, N \mod p_i$ can be computed for every i. The only message that Alice receives is $z \stackrel{\triangle}{=} z_1^{Q_b} \cdot z_2^{P_b} \cdot z_3 \mod m$. By the definition of z_3 and the encryption system $z_3 = y^{P_bQ_b}u^t \mod m$ where u is a random in Z_m^* . Thus z is a random element in the appropriate coset (all the elements whose decryption is $N - P_aQ_a \mod t$). \square

Lemma 6. The running time of the protocol is dominated by the single decryption Alice executes, the communication complexity is 3σ and the protocol requires one round of communication.

5 Amortization and Initial Primality Test

The initial primality test consists of checking whether a candidate N is divisible by one of the first k primes p_1, \ldots, p_k . If it is then either $P = P_a + P_b$ or $Q = Q_a + Q_b$ is not a prime. This test can be carried out by Alice following the computation of N.

If a candidate N passes the initial primality test, Alice publishes its value and it becomes a candidate for the full primality test of [4]. However, if it fails the test a new N has to be computed. In this section we show how to efficiently find a new

candidate following a failure of the initial test by the previous candidate. The total cost of computing a series of candidates is lower than using the protocols of section 4 each time anew. We show two different approaches. One for the oblivious transfer and oblivious polynomial evaluation protocols, and the other for the homomorphic encryption protocol.

5.1 OT and Oblivious Polynomial Evaluation

Suppose that after Alice and Bob discard a certain candidate N they compute a new one by having Alice retain the previous P_a, Q_a and having Bob choose new P_b, Q_b . In that case, as we show below, computing the new N can be much more efficient than if both parties choose new shares. The drawback is that given both values of N Bob can gain information about P_a, Q_a . Therefore, in this stage (unlike the full primality test) Alice does not send the value of N to Bob.

Assume Bob holds two sequences of strings: (a_1^0, \ldots, a_n^0) , (a_1^1, \ldots, a_n^1) and Alice wishes to retrieve one sequence without revealing which one to Bob and without gaining information about the second sequence. Instead of invoking a $\binom{2}{1}$ -OT protocol n times the players agree on a pseudo-random generator G and do the following:

- 1. Bob chooses two random seeds s_1, s_2 .
- 2. Alice uses a single invocation of $\binom{2}{1}$ -OT to gain s_b , where $b \in \{0, 1\}$ denotes the desired string sequence.
- 3. Bob sends to Alice the first sequence masked (i.e bit by bit exclusive-or) by $G(s_1)$ and the second sequence masked by $G(s_2)$.

Alice can unmask the required sequence while the second sequence remains pseudo-random. In the protocol of subsection 4.1 N is computed using only oblivious transfers in which Alice retrieves a set of 2σ strings from Bob. Alice's choices of which strings to retrieve depend only on her input P_a, Q_a . Therefore if Alice retains P_a and Q_a while Bob selects a sequence of inputs $(P_b^1, Q_b^1), \ldots, (P_b^n, Q_b^n)$, the two players can compute a sequence of candidates N^1, \ldots, N^n with as many oblivious transfers as are needed to compute a single N.

The same idea can be used in the oblivious polynomial evaluation protocol, as noted in [19]. The evaluation of many polynomials at the same point requires as many oblivious transfers as the evaluation of a single polynomial at that point. Thus, computing a sequence of candidates N requires only $2 \log m(2d_{Q,x} + 1)$ computations of $\binom{2}{1}$ -OT.

5.2 Homomorphic Encryption

Alice and Bob combine the two stages of computing N and trial divisions by using the protocol of subsection 4.3 flexibly. Let $p_1, \ldots, p_{j'}$ be the j' smallest distinct primes such that $\pi_{i=1}^{j'} p_i > 2^{(\sigma-1)/2}$. Alice and Bob pick their elements at random in the range $0, \ldots, \pi_{i=1}^{j'} p_i - 1$ by choosing random elements in each \mathcal{Z}_{p_i} for $i = 1, \ldots, j'$.

Alice and Bob compute $N \mod p_i$ as described in subsection 4.3. If $N \equiv 0 \mod p_i$ then at least one of the elements $P = P_a + P_b$ or $Q = Q_a + Q_b$ is divided by p_i . In that case, Alice and Bob choose new, random elements: $P_a, P_b, Q_a, Q_b \mod p_i$, and recompute $N \mod p_i$. The probability of this happening is less than $2/p_i$. Thus the expected number of re-computations is less than $\sum_{i=1}^{j'} 2/p_i$. This quantity is about 3.1 for $\sigma = 1024$ (2 does not cause a problem because $P_a \equiv Q_a \equiv 3 \mod 4$ and $P_b \equiv Q_b \equiv 0 \mod 4$).

Setting $P_a \mod p_i$ for $i=1,\ldots,j'$ determines P_a , and by the same reasoning the other 3 shares that Alice and Bob hold are also set. The two players complete the computation of N by determining the value of N mod p_i (using the protocol of subsection 4.3) for $i=j'+1,\ldots,j$, where $\pi_{i=1}^j p_i > 2^{\sigma}$. If for one of these primes $N \equiv 0 \mod p_i$ Alice and Bob discard their shares and pick new candidates p_i

6 Computing d

Alice and Bob share $\phi(N)$ in an additive manner. Alice holds $\phi_a \stackrel{\triangle}{=} N - P_a - Q_a + 1$, Bob holds $\phi_b = -Q_b - P_b$ and $\phi_a + \phi_b = \phi(N)$. The two parties agree on a public exponent e. Denote $\eta \stackrel{\triangle}{=} \lceil \log e \rceil$. We follow in the footsteps of the Boneh-Franklin protocol and employ their algorithm to invert e modulo $\phi(N)$ without making reductions modulo $\phi(N)$:

- 1. Compute $\zeta = -\phi(N)^{-1} \mod e$.
- 2. Compute $d = (\zeta \phi(N) + 1)/e$.

Now $de \equiv 1 \mod \phi(N)$ and therefore d is the inverse of e modulo $\phi(N)$.

As a first step Alice and Bob change the additive sharing of $\phi(N)$ into a multiplicative sharing modulo e, without leaking information about $\phi(N)$ to either party. At the end of the sub-protocol Alice holds $r\phi(N)$ mod e and Bob holds r^{-1} mod e, where r is a random element in \mathbb{Z}_e^* .

- 1. Bob chooses uniformly at random $r \in \mathcal{Z}_e^*$. Alice and Bob invoke the protocol of subsection 4.1, setting $\mathcal{R} \stackrel{\triangle}{=} \mathcal{Z}_e$, $a \stackrel{\triangle}{=} \phi_a$ and $b \stackrel{\triangle}{=} r$. At the end of the protocol Alice holds x and Bob holds y such that $x + y \equiv \phi_a r \mod e$.
- 2. Bob sends $y + \phi_b r \mod e$ to Alice.
- 3. Alice computes $x + y + \phi_b r \equiv r\phi(N) \mod e$, and Bob computes $r^{-1} \mod e$.

Lemma 7. The computation time and the communication complexity of the protocol are dominated by η oblivious transfers.

After completing the sub-protocol we described above, Alice and Bob finish the inversion algorithm by performing the following steps:

An interesting optimization is not to discard the whole share (P_a, P_b, Q_a, Q_b) , but for each specific share, say P_a , only to select a new $P_a \mod p_i$ for $i = j' - c, \ldots, j'$, where c is a small constant. The probability is very high that the new N thus defined is not a multiple of p_i .

- 1. The two parties hold multiplicative shares of $\phi(N)$ mod e. They compute the inverse of their shares modulo e and thus have ζ_a, ζ_b respectively such that $\zeta_a \cdot \zeta_b \equiv -\phi(N)^{-1} \equiv \zeta \mod e$.
- 2. Alice and Bob re-convert their current shares into additive shares modulo e, i.e ψ_a, ψ_b such that $\psi_a + \psi_b \equiv \zeta \mod e$. Bob chooses randomly $\psi_b \in \mathcal{Z}_e$ and the two parties combine to enable Alice to gain $\psi_a \stackrel{\triangle}{=} -\psi_b + \zeta_a \zeta_b \mod e$. This is done by employing essentially the same protocol we used for transforming ϕ_a, ϕ_b into a multiplicative sharing. If we replace ζ_a by ϕ_a, ζ_b by r and $-\psi_b$ by $r\phi_b$ we get the same protocol.
- 3. The two parties would like to compute $\zeta\phi(N)$. What they actually compute is $\alpha \stackrel{\triangle}{=} (\psi_a + \psi_b)(\phi_a + \phi_b)$. The result is either exactly $\zeta\phi(N)$ or $(\zeta + e)\phi(N)$. The computation is carried out similarly to the computation of N in subsection 4.1. The ring used is \mathcal{Z}_k where $k > 4e \cdot 2^{\sigma}$. We modify the protocol in two ways. The first modification is that α is not revealed to Alice but remains split additively over \mathcal{Z}_k among the two players. In other words they perform step 1 of the protocol in subsection 4.1 and additively share $\psi_a\phi_b + \psi_b\phi_a$. Alice adds $\psi_a\phi_a$ to her share and Bob adds $\psi_b\phi_b$ to his.

The sum of the two shares over the integers is either α or $\alpha + k$. The second option is unacceptable (unlike the two possibilities for the value of α). In order to forestall this problem we introduce a second modification. The sharing of α results in Alice holding $\alpha + y'$ mod k and Bob holding -y' mod k. Furthermore Bob selects y' by his random choices. We require Bob to make those choices subject to the condition that y' < k/2. Since y' is not completely random, Alice might gain a slight advantage. However, that advantage is at most a single bit of knowledge about $\phi(N)$ (which can be guessed in any attempt to discover $\phi(N)$).

4. Alice sets $d_a \stackrel{\triangle}{=} \lceil (\alpha + y')/e \rceil$ and Bob sets $d_b \stackrel{\triangle}{=} \lfloor (-y'+1)/e \rfloor$ (these calculations are over the integers). Hence, either $d_a + d_b = (\zeta \phi(N) + 1)/e$ or $d_a + d_b = (\zeta \phi(N) + 1)/e + \phi(N)$.

7 Improvements

In this section we suggest some efficiency improvements.

Off-line preprocessing: The performance of the protocol based on Benaloh's encryption can be significantly improved by some off-line preparation. Obviously, for any t used as a modulus in the protocol a suitable encryption system has to be constructed (i.e a suitable m = pq has to be found, a table of all values $T_M = y^{M\phi(m)/t} \mod m$ has to be computed etc.).

Further improvement of the online communication and computational complexity can be attained at the cost of some space and off-line computation. Instead of constructing a separate encryption system for t_1 and t_2 , Alice constructs a single system for $t=t_1t_2$. The lookup table needed for decryption is formed as follows. Alice computes $T_M=y^{M\phi(m)/t} \mod m$ for $M=0,\ldots,t_1-1$ and $T_{\bar{M}}=y^{\bar{M}t_1\phi(m)/t} \mod m$ for $\bar{M}=0,\ldots,t_2-1$. The entries of the table are

obtained by calculating $T_M T_{\bar{M}}$ mod m for every pair M, \bar{M} . Constructing this table takes more time than constructing the two separate tables for t_1, t_2 . The additional time is bounded by the time required to compute $t_2(t_1 + \log t_1 \log t_2)$ modular multiplications over \mathcal{Z}_m (computing $T_{\bar{M}}$ involves $\log t_1 \log t_2$ multiplications in comparison with $\log t_2$ multiplications in the original table).

The size of the table is $t \log m$ (slightly more than $t\sigma$). This figure which might be prohibitive for large t can be significantly reduced. After computing every entry in the table it is possible by using perfect hashing [14] to efficiently generate a 1-to-1 function h from the entries of the table to $0, \ldots, 3t-1$. A new table is now constructed in which instead of original entry T_M an entry $(h(T_M), M)$ is stored. Decryption of z is performed by finding the entry holding $h(z^{\phi(m)/t})$ mod m and reading the corresponding M. The size of the stored table is $2t \log t$. As an example of the reduction in space complexity consider the case $t=3\cdot 751=2253$. The original table requires more than 2^{21} bits while the hashing table requires less than 2^{16} bits.

It is straightforward to use $t = t_1t_2$ instead of t_1 and t_2 separately in subsections 4.3 and 5.2. The protocols in both subsections remain almost without change apart from omitting the sub-protocols for t_1 and t_2 and adding a subprotocol for t. In subsection 5.2 it is not enough to check whether $N \equiv 0 \mod t$. It is necessary to retain the two tests of $N \mod t_1$ and $N \mod t_2$.

Note that here we need the stronger *higher residuosity* intractability assumption replaces the *prime residuosity* assumption.

Alternative computation of d: The last part of generating RSA keys is constructing the private key. Using Benaloh's encryption we can sometimes improve the computation and communication complexity of the construction in comparison with the results of section 6. The improvement is possible if the parameter t of a Benaloh encryption can be set to e (that is, the homomorphism is modulo e) so that efficient decryption is possible. Therefore, e has to be a product of "small" primes, see [3]. The protocol for generating and sharing d is a combination of the protocols of subsection 4.3 and section 6. We leave the details to the full version of the paper.

8 Performance

The most resource consuming part of our protocol, in terms of computation and communication, is the computation of N together with trial divisions. We use trial divisions because of the following result by DeBruijn [10]. If a random $\sigma/2$ bit integer passes the trial divisions for all primes less than B then asymptotically:

$$\Pr[p \text{ prime } | p \not\equiv 0 \bmod p_i, \forall p_i \leq B] = 5.14 \frac{\ln B}{\sigma} (1 + o(\frac{2}{\sigma})).$$

We focus on the performance of the more efficient version of our protocol, using homomorphic encryption. We also assume that the off-line preprocessing suggested in section 7 is used. Let j', j be defined as in section 4.3. We pair off the first j' primes and prepare encryption systems for products of such pairs (as

in section 7.) The number of exponentiations (decryptions) needed to obtain one N is on average about j+3-j'/2. The probability that this N is a product of two primes is approximately $(5.14 \ln p_{j'}/\sigma)^2$.

Another obvious optimization is to divide the decryptions between the two parties evenly. In other words for half the primes the first party plays Alice and for the other half they switch.

If $\sigma=1024$ and we pair off the first j'=76 primes the running time of our protocol is (by a rough estimate) less than 10 times the running time of the Boneh-Franklin protocol. The communication complexity is (again very roughly) 42MB. If the participants are willing to improve the online communication complexity in return for space and pair off the other j-j' needed to compute N the communication complexity is reduced to about 29MB.

Open problem: Boneh and Franklin show in [4] how to test whether N is a product of two primes, where both parties hold N. It would be interesting to devise a distributed test to check whether N is a product of two primes if Alice holds N, P_a, Q_a and Bob only has his private shares P_b, Q_b . The motivation is that in the oblivious transfer and oblivious polynomial evaluation protocols we presented P_a, Q_a will have to be selected only once. Thus the number of oblivious transfers in the whole protocol is reduced to the number required for computing a single candidate N.

Acknowledgments: I'd like to thank the anonymous referees for several enlightening comments, Benny Chor for suggesting this area of research, Josh Benaloh for his help and Yuval Ishai for his usual insights.

References

- M. Ben-OR, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computation. In *Proc. of* 20th STOC, pages 1–10, 1988.
- 2. J. Benaloh. Verifiable Secret-Ballot Elections. PhD thesis, Yale University, 1987.
- J. Benaloh. Dense probabilistic encryption. In Proc. of the Workshop on Selected Areas of Cryptography, pages 120–128, May 1994.
- 4. D. Boneh and M. Franklin. Efficient generation of shared rsa keys. In *Proc. of Crypto 97*, pages 425–439. Springer-Verlag, 1997. THe full version appears on the web at theory.stanford.edu/ dabo/pubs.html.
- C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology- EUROCRYPT* '99, 1999.
- 6. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstracts). In Proc. of 20^{th} STOC, pages 11–19, 1988.
- B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In Proc. of 36th FOCS, pages 41–50, 1995.
- 8. C. Cocks. Split knowledge generation of rsa parameters. In M. Darnell, editor, Cryptography and Coding, 6th IMA international conference, pages 89–95, December 1997. Appears on the web at www.cesg.gov.uk/downlds/math/rsa.pdf.

- 9. C. Cocks. Split generation of rsa parameters with multiple participants, 1998. Appears on the web at www.cesg.gov.uk/downlds/math/rsa2.pdf.
- N. DeBruijn. On the number of uncancelled elements in the sieve of eratosthenes. Proc. Neder. Akad., 53:803–812, 1950. Reviewed in Leveque Reviews in Number Theory, Vol. 4, Section N-28, p. 221.
- A. DeSantis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securly. In Proc. of STOC, pages 522–533, 1994.
- Y. Desmedt. Threshold cryptography. European Transactions on Telecommunications and Related Technologies, 5(4):35–43, July-August 1994.
- Y. Frankel, P. D. MacKenzie, and M. Yung. Robust efficient distributed rsa-key generation. In Proc. of 30th STOC, pages 663-672, 1998.
- 14. M. Fredman, J. Komlos, and E. Szemeredi. Storing a sparse table in o(1) worst case access time. *Journal of the ACM*, 31:538–544, 1984.
- Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In Proc. of 30th STOC, 1998.
- O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game (extended abstract). In Proc. of 19th STOC, pages 218–229, 1987.
- S. Goldwasser and S. Micali. Probabilistic encryption. Journal of Computer and systems science, 28:270–299, 1984.
- E. Kushilevitz and R. Ostrovsky. Single-database computationally private information retrieval. In Proc. of 38th FOCS, pages 364–373, 1997.
- 19. M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of 31st STOC*, pages 245–254, 1999.
- Guillaume Poupard and Jacques Stern. Generation of shared rsa keys by two parties. In ASIACRYPT'98, pages 11–24. Springer-Verlag LNCS 1514, 1998.
- 21. Julien P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In ASIACRYPT'98, pages 357–371. Springer-Verlag, 1998.
- A.C. Yao. How to generate and exchange secrets. In Proc. of 27th FOCS. IEEE Press, 1986.

Robust Distributed Multiplication without Interaction

Masayuki Abe

NTT Laboratories Nippon Telegraph and Telephone Corporation 1-1 Hikari-no-oka Yokosuka-shi Kanagawa-ken, 239-0847 Japan abe@isl.ntt.co.jp

Abstract. An optimally resilient distributed multiplication protocol that enjoys the property of non-interactivity is presented. The protocol relies on a standard cryptographic assumption and works over a complete, synchronous, untappable network with a broadcast channel. As long as no disruption occurs, each player uses those channels only once to send messages; thus no interaction is needed among players. The cost is an increase in local computation and communication complexity that is determined by the factor of the threshold.

As an application of the proposed protocol we present a robust threshold version of the Cramer-Shoup cryptosystem, which is the first non-interactive solution with optimal resilience.

Keywords: Distributed Multiplication, Round Complexity, Multi-Party Computation, Cramer-Shoup Cryptosystem

1 Introduction

Background: Practical cryptographic applications sometimes put their fundamental trust on one entity such as a trusted authority or a tamper-proof device. When such a trustee is not preferred, one practical solution is to distribute the trust among several parties in order to make the system robust against the leakage of partial secrets or Byzantine faults [20,12]. Secret sharing [24] allows a party to keep a secret in a distributed manner so that some application-dependent functions are computed in collaboration without revealing the secret. So far, much work has been devoted to the investigation of such multi-party computation, e.g., [18,2,6,23,1]. In particular, some recent works realized practical multi-party versions of public-key cryptographic functions such as decryption and signature generation, e.g., [10,15,8], key generation, e.g., [21,3,14,16] and so on. Sometimes they are called "function sharing" or "threshold cryptography."

One of the basic ingredients for all robust threshold schemes is verifiable secret sharing, e.g., [11,22]. A useful property of verifiable secret sharing schemes is that they allow to compute linear combination of shared secrets without any communication between players. Multiplication on shared secrets, however, remains a cumbersome process because it requires several interactions among players in

order to keep the threshold unchanged. Since round complexity can be the dominant factor determining efficiency, especially when the protocol is implemented over a network where communication overhead can not be ignored, it is of practical interest to construct a distributed multiplication protocol that enjoys less round complexity and yet remains practical in computation and communication.

Related Work: A generic approach for evaluating any boolean circuit in a cryptographic multi-party setting was shown in [18], which assumes the use of a probabilistic encryption scheme. Although the result is theoretically significant, this approach will turn out to be inefficient if it is directly applied to construct a multi-party protocol for a particular function such as multiplication in a large field.

In [2], Ben-Or, Goldwasser and Wigdersen introduced the degree reduction technique that converts shares on a 2t-degree polynomial into ones on a t-degree truncated polynomial. For n players, the technique tolerates less than n/3 corrupted players. The scheme shown in [23] tolerates less than n/2 faulty players and also follows this line. The scheme in [6] relies on the massive use of zero knowledge proofs. All the schemes that do not rely on computational assumptions require far too many interactions among players, even though some of them are constant-round. Beaver improved the scheme in [23] but the round complexity remains unchanged [1].

M. Rabin invented a simple (non-robust) distributed multiplication scheme in [17], which can be replaced with the previous degree reduction technique. In the information theoretic setting, Cramer et al. added robustness to Rabin's basic scheme by developing new Information Checking technique [7]. In the cryptographic setting, Gennaro et al., provided robustness by having each player invoke a four-move zero-knowledge interactive proof [17]. All players can be logically combined into a single verifier (by using an information theoretically secure bit commitment scheme and a verifiable secret sharing), and if this combination performs all zero-knowledge proofs in parallel, one can obtain a four-move protocol for distributed multiplication. The immediately thought is that the public-coin interactive zero-knowledge protocol can be turned into an non-interactive one by using Fiat-Shamir heuristics [13] at the cost of losing provable security in a standard cryptographic model. Cerecedo et al., introduced a dedicated protocol for multiplying a distributedly-generated random number and a shared secret in a construction of shared signature schemes [5]. Their scheme works non-interactively for thresholds under one third of the players, and two additional accesses to the communication channels are needed to obtain optimal resiliency. Their schemes are not information-theoretically secure, and hence leak some information about shared secrets. Indeed, the attack introduced by Gennaro et al., in [16] is applicable.

Our Contribution: We provide a distributed multiplication protocol that works in a cryptographic setting and offers the following properties:

- Non-interactivity: As long as no disruption occurs, players access private network and broadcast channel only once to send data without needing to synchronize to other players. The players agree on correctness implicitly by silence (or explicitly by broadcasting 1 bit message if needed). No zeroknowledge proof is used in the protocol.
- Provably secure in a standard model: The security can be proved under the intractability assumption of the discrete logarithm problem. We do not assume the use of shared random string or random oracles.
- Information theoretic secrecy: The shared secrets are secure against infinitely powerful adversaries.
- Optimal resiliency: Tolerate the corruption of less than half of players.

The cost of achieving these properties is an increase in computation and communication complexity. Our scheme suffers $\mathcal{O}(t)$ increase in computation and communication complexity, where t is the threshold, compared to the above mentioned four-move scheme by Gennaro et al. Yet it is comparable to the scheme by Cerecedo et al. in both computation and communication costs. More concretely, our scheme consumes about three to four times more computation and communication than that of the random number generation scheme by Pedersen [22], which is often used in threshold cryptography.

Our result will reduce the round complexity of several cryptographic applications, e.g., all variants of El Gamal signature scheme that have embedded multiplication, and threshold key generation for signature or encryption schemes that use a composite modulus like RSA.

We use our protocol to construct a robust threshold version of the Cramer-Shoup cryptosystem that withstands a minority of corrupt decryption servers. The protocol is non-interactive except for precomputation for randomizing decryption keys. A threshold Cramer-Shoup cryptosystem with optimal resiliency was described by Canetti et al. in [4] that uses four-move interactive zero-knowledge proofs. One of their variants has the non-interactivity property, but it only tolerates $\mathcal{O}(\sqrt{n})$ corrupted servers. Accordingly, we show the first optimally resilient and non-interactive robust threshold cryptosystem that is secure against adaptive chosen ciphertext attacks under standard cryptographic assumptions.

Organization: Section 2 sketches the model and our goal. Section 3 presents the proposed multiplication protocol. Proof of security is given in Section 4. Section 5 extends the protocol to remove an assumption about the inputs. In Section 6, we construct a threshold version of the Cramer-Shoup cryptosystem. Some concluding remarks are given in Section 7.

2 Model

2.1 Setting

Communication channels: Our protocol assumes a *synchronous private net-work* where a message is assured of being delivered in a fixed period. The network

is assumed to be secure and complete, that is, every pair of players is connected by an untappable and mutually authenticated channel. Furthermore, we assume the use of a *broadcast channel* where all players receive the same information sent from authenticated players.

Players and an adversary: Let \mathcal{P} be a set of players such that $\mathcal{P} = \{1, \ldots, n\}$. Player $i \in \mathcal{P}$ is assumed to be a probabilistic polynomial-time Turing machine. Among those players, there are up to t corrupt players completely controlled by a static adversary.

Computational assumption: We use large primes p and q that satisfy q|p-1. G_q denotes a multiplicative subgroup of degree q in Z_p . It is assumed that solving the discrete logarithm problem in G_q is intractable. All arithmetic operations are done in Z_p unless otherwise stated.

2.2 Notations for Verifiable Secret Sharing

Our scheme uses an information theoretically secure verifiable secret sharing scheme by Pedersen [22]. Let g and h be elements of G_q where $\log_g h$ is unknown. To share secret S in Z_q , a dealer first chooses two t-degree random polynomials f(X) and d(X) from $Z_q[X]$ such that f(0) = S is satisfied. Let R denote random free term of d(X). The dealer sends a pair $(S_i, R_i) := (f(i), d(i)) \in Z_q^2$ to player i via a private channel. The dealer then broadcasts $E_k := g^{a_k} h^{b_k}$ for $k = 0, \ldots, t$ where a_k and b_k are coefficients of k-degree term of f(X) and d(X) respectively. Note that S and R are committed to E_0 as $E_0 = g^S h^R$. Correctness of a share (S_i, R_i) can be verified by checking the relation

$$g^{S_i} h^{R_i} = \prod_{k=0}^{t} E_k^{i^k}.$$
 (1)

We may refer the right side of the above verification predicate as *verification* commitment for player i. Note that any player can compute the verification commitments without knowing the corresponding shares. Hereafter, we denote the execution of this verifiable secret sharing protocol by

$$PedVSS(S, R)[g, h] \xrightarrow{f, d} (S_i, R_i)(E_0, \dots, E_t).$$

Polynomials put on an arrow may be omitted if no misunderstanding is expected. Secret S is reconstructed by interpolation as in

$$S := \sum_{i \in \mathcal{Q}} \lambda_{i,\mathcal{Q}} S_i \bmod q, \quad \text{where} \quad \lambda_{i,\mathcal{Q}} := \prod_{j \in \mathcal{Q}, j \neq i} \frac{j}{j-i} \bmod q \tag{2}$$

where Q is any set of at least t+1 qualified players whose share (S_i, R_i) satisfies Equation 1.

Lemma 1. The above verifiable secret sharing scheme offers the following properties:

- 1. If all players follow the protocol, shares are accepted with probability 1.
- 2. For any $Q \subseteq \mathcal{P}$ of size no less than t+1, a set of shares $(S_i, R_i) \mid i \in Q$ each of which satisfies Equation 1 recovers (S, R) that satisfies $E_0 = g^S h^R$ with overwhelming probability in |q|.
- 3. Let view_A be a unified view in a protocol run obtained by corrupted players in A. For any $A \subset \mathcal{P}$ of size at most t, for any secret $S \in Z_q$, and for any $x \in Z_q$,

$$Prob[S = x | view_{\mathcal{A}}] = Prob[S = x].$$

As it is well-known, the players can share a random number unknown to any set of players less than t+1 by using PedVSS. We refer to this shared random number generation protocol as

$$RandVSS([S], [R])[g, h] \rightarrow (S_i, R_i)(E_0, \dots, E_t).$$

Brackets in parentheses imply that the numbers are unknown to any set of players less than the threshold.

2.3 Goal

Our goal is to construct a protocol such that players in \mathcal{P} eventually share the product of $A \in \mathbb{Z}_q$ and $B \in \mathbb{Z}_q$ as if an honest dealer executes

$$PedVSS(A \cdot B, R^c)[g, h] \rightarrow (C_i, R_i^c)(EC_0, \dots, EC_t)$$
 (3)

for some random number $R^c \in \mathbb{Z}_q$. We assume that A and B have been already shared by an honest dealer as

$$PedVSS(A, R^a)[g, h] \rightarrow (A_i, R_i^a)(EA_0, \dots, EA_t)$$
 (4)

$$PedVSS(B, R^b)[g, h] \rightarrow (B_i, R_i^b)(EB_0, \dots, EB_t)$$
 (5)

where R^a and R^b are random numbers taken uniformly from Z_q .

3 Protocol

The main idea to reduce round complexity is to use VSS instead of ZKP to confirm the relationship of committed values. Recall that if a dealer completes $PedVSS(S,R) \rightarrow (S_i,R_i)(E_0,\ldots,E_t)$ correctly, it implies that the dealer knows S and R committed to $E_0 = g^S h^R$. Conversely, if the dealer does not know the representation of E_0 , he can not complete the protocol successfully. Similarly, if P_i completes two PedVSS executions such as

$$PedVSS(S,R) \rightarrow (S_i, R_i)(E_0, \dots, E_t),$$

 $PedVSS(S', R') \rightarrow (S'_i, R'_i)(E'_0, \dots, E'_t)$

and if $S_i = S'_i$ holds for at least t + 1 evaluation points, it proves equality of committed values, that is, S = S'. Those arguments are derived immediately from Lemma 1. Furthermore, if P_i completes three executions,

$$PedVSS(S, R) \to (S_i, R_i)(E_0, \dots, E_t),$$

 $PedVSS(S', R') \to (S'_i, R'_i)(E'_0, \dots, E'_t),$
 $PedVSS(S'', R'') \to (S''_i, R''_i)(E''_0, \dots, E''_t),$

and $S_i + S'_i = S''_i$ holds for more than t + 1 *i*-s, it implies additive relation of committed values, that is, S + S' = S''. Multiplicative relation, which is our main interest, can be proved in a rather tricky way as is shown in the main protocol.

Our protocol is constructed over the simplified multiplication protocol in [17], which is reviewed below. This protocol works with t up to t < n/2.

[Protocol: Basic Multiplication]

BM-1: Each player i picks a t-degree random polynomial to share $A_i \cdot B_i$. The share C_{ij} is privately sent to player $j, j \in \mathcal{P}$.

BM-2: Each player j computes his share C_j for $A \cdot B$ as

$$C_j := \sum_{i \in \mathcal{P}} \lambda_{i,\mathcal{P}} C_{ij} \mod q.$$

[End]

Let Q_2 and Q_1 be subsets of \mathcal{P} whose sizes are 2t+1 and t+1 respectively. Recall that $A \cdot B$ can be recovered from $A_i \cdot B_i$ as $A \cdot B = \sum_{i \in Q_2} \lambda_{i,Q_2} A_i \cdot B_i$. Since the above protocol allows players to recover $A_i \cdot B_i$ by computing $A_i \cdot B_i = \sum_{j \in Q_1} \lambda_{j,Q_1} C_{ij}$, it holds that $A \cdot B = \sum_{i \in Q_2} \lambda_{i,Q_2} \sum_{j \in Q_1} \lambda_{j,Q_1} C_{ij} = \sum_{j \in Q_2} \lambda_{j,Q_2} C_j$. Therefore, $A \cdot B$ can be recovered from any set of more than t+1 C_j -s.

To add robustness to the basic multiplication protocol, each player must convince other players that C_{ij} is a share of $A_i \cdot B_i$. Let the verification commitment for (A_i, R_i^a) and (B_i, R_i^b) be $VA_i := \prod_{k=0}^t EA_k^{i^k}$ and $VB_i := \prod_{k=0}^t EB_k^{i^k}$ respectively.

[Protocol: Robust Distributed Non-interactive Multiplication]

DM-1: Each player i randomly picks t-degree polynomials f_1 , d_1 and d_2 from $Z_q[X]$ so that $f_1(0) = A_i$ and $d_1(0) = R_i^a$ are satisfied. Let R_i^{ab} denote a randomly chosen free term of $d_2(X)$. Player i shares A_i twice as

$$PedVSS(A_i, R_i^a)[g, h] \xrightarrow{f_1, d_1} (A_{ij}, R_{ij}^a)(\langle VA_i \rangle, EA_{i1}, \dots, EA_{it}), \text{ and}$$

 $PedVSS(A_i, R_i^{ab})[VB_i, h] \xrightarrow{f_1, d_2} (\langle A_{ij} \rangle, R_{ij}^{ab})(EAB_{i0}, \dots, \text{it EAB}_{it}).$

Angle brackets mean that the variable can be locally computed by the receivers, or it has been sent before. Note that $VB_i (= g^{B_i} h^{R_i^b})$ is used as the base of the commitments in the second sharing. Player i then selects two random polynomials $f_2(X)$ and $d_3(X)$ that satisfy $f_2(0) = A_i \cdot B_i \mod q$ and $d_3(0) = R_i^b \cdot A_i + R_i^{ab} \mod q$, and performs

$$PedVSS(A_i \cdot B_i, R_i^b \cdot A_i + R_i^{ab})[g, h] \xrightarrow{f_2, d_3} (C_{ij}, R_{ij}^c)(\langle EAB_{i0} \rangle, EC_{i1}, ..., EC_{it}).$$

DM-2: Each player j verifies everything received from player i as

$$g^{A_{ij}}h^{R_{ij}^a} = VA_i \prod_{k=1}^t EA_{ik}^{j^k},$$
 (6)

$$VB_{i}^{A_{ij}}h^{R_{ij}^{ab}} = \prod_{k=0}^{t} EAB_{ik}^{j^{k}},$$
(7)

$$g^{C_{ij}}h^{R_{ij}^c} = EAB_{i0} \prod_{k=1}^t EC_{ik}^{j^k}.$$
 (8)

If a check fails, player j declares so and goes to the disqualification protocol (see below).

DM-3: Let \mathcal{I} be a set of qualified players in **DM-2**. $|\mathcal{I}| \geq 2t + 1$ must hold. Each player j in \mathcal{I} computes

$$C_j := \sum_{i \in \mathcal{I}} \lambda_{i,\mathcal{I}} C_{ij} \bmod q, \tag{9}$$

$$R_j^c := \sum_{i \in \mathcal{I}} \lambda_{i,\mathcal{I}} R_{ij}^c \bmod q, \tag{10}$$

$$EC_k := \prod_{i \in \mathcal{I}} EC_{ik}^{\lambda_{i,\mathcal{I}}} \text{ for } k = 0, \dots, t,$$
(11)

where $EC_{i0} = EAB_{i0}$.

[End]

Suppose that player j finds that the shares sent from player i do not satisfy all of the verification equations in **DM-2**. All players then perform the following protocol.

[Protocol: DISQUALIFICATION]

DQ-1: Player i is requested to broadcast all the data that he privately sent to player j, which is $(A_{ij}, R^a_{ij}, R^{ab}_{ij}, C_{ij}, R^c_{ij})$. (If player i keeps silent, he is disqualified immediately.)

DQ-2: If t+1 or more players conclude that the shares satisfy the verification equations in **DM-2**, player j must accept the shares just published. On the other hand, if t+1 or more players decide that those shares are faulty, player i is disqualified.

[End]

Let \mathcal{D} be a set of disqualified players. If $|\mathcal{P} \setminus \mathcal{D}| < 2t + 1$ happens because of the disqualification, the remaining players in $\mathcal{P} \setminus \mathcal{D}$ have to recover (A_i, R_i^a) and (B_i, R_i^b) owned by a disqualified player i. For this to be done, we use the *share recovery* protocol which has been used in proactive secret sharing [19]. Note that the protocol can be completed in a robust and secure way if at least t+1 honest players exist. $(A_i \text{ and } R_i^a \text{ can be recovered directly from shares } (A_{ij}, R_{ij}^a)$ if more than t+1 correct shares exist.)

4 Security

Lemma 2 (Correctness). If all players follow the protocol, every player i in \mathcal{P} obtains a share (C_i, R_i^c) and commitments (EC_0, \ldots, EC_t) that satisfy $g^{C_i} h^{R_i^c} = \prod_{k=0}^t EC_k^{i^k}$ and $A \cdot B = \sum_{i \in \mathcal{Q}} \lambda_{i,\mathcal{Q}} C_i \mod q$ for any $\mathcal{Q} \subseteq \mathcal{P}$ of size no less than t+1

Proof. According to Lemma 1, any set of no less than t+1 correct (A_{ij}, R^a_{ij}) that passes verification 6 recovers (A_i, R^a_i) that satisfies $g^{A_i} h^{R^a_i} = VA_i$. Similarly, any set of no less than t+1 correct (A_{ij}, R^{ab}_{ij}) that passes verification 7 recovers (\tilde{A}_i, R^{ab}_i) that satisfies $VB^{\tilde{A}_i} h^{R^{ab}_i} = EAB_{i0}$. Because both A_i and \tilde{A}_i are recovered from the same shares A_{ij} , we have $\tilde{A}_i = A_i$. Therefore,

$$EAB_{i0} = VB_i^{A_i} h^{R_i^{ab}} = g^{A_i \cdot B_i} h^{A_i \cdot R_i^b + R_i^{ab}}.$$

Observe that EAB_{i0} is also used to verify (C_{ij}, R_{ij}^c) with g and h. Hence, for any $Q \subseteq \mathcal{P}$ of size no less than t+1, it holds that $A_i \cdot B_i = \sum_{j \in Q} C_{ij} \lambda_{j,Q} \mod q$. Recall that $A \cdot B$ can be recovered from a set of correct $A_i \cdot B_i$ as $A \cdot B = \sum_{i \in \mathcal{I}} (A_i \cdot B_i) \lambda_{i,\mathcal{I}} \mod q$ holds for any $\mathcal{I} \subseteq \mathcal{P}$ of size no less than 2t+1. Thus,

$$\sum_{j \in \mathcal{Q}} C_j \lambda_{j,\mathcal{Q}} = \sum_{j \in \mathcal{Q}} (\sum_{i \in \mathcal{I}} C_{ij} \lambda_{i,\mathcal{I}}) \lambda_{j,\mathcal{Q}}$$

$$= \sum_{i \in \mathcal{I}} (\sum_{j \in \mathcal{Q}} C_{ij} \lambda_{j,\mathcal{Q}}) \lambda_{i,\mathcal{I}}$$

$$= \sum_{i \in \mathcal{I}} (A_i \cdot B_i) \lambda_{i,\mathcal{I}}$$

$$= A \cdot B \pmod{q}$$

holds for any $Q \subseteq \mathcal{P}$ of size no less than t+1.

Lemma 3 (Secrecy). Let A be $A \subset P$ such that $|A| \leq t$. Let $view_A$ be unified view of players in A during a protocol run with secrets (A, R^a) and (B, R^b) . Similarly, let $view'_A$ be a view during a protocol run with (\tilde{A}, \tilde{R}^a) and (\tilde{B}, \tilde{R}^b) that satisfy $g^{\tilde{A}} h^{\tilde{R}^a} = g^A h^{R^a}$ and $g^{\tilde{B}} h^{\tilde{R}^b} = g^B h^{R^b}$. Then, $view_A$ and $view'_A$ are perfectly indistinguishable.

Proof. We prove the above statement by constructing a protocol simulator that works without knowing the secrets committed to EA_0 and EB_0 . Let $\mathcal{H} := \mathcal{P} \setminus \mathcal{A}$. For simplicity, we exclude the disqualification steps for a while. The view $view_{\mathcal{A}}$ in a protocol run with secrets A, B can be divided into the following two parts:

$$view_{\mathcal{A},private} = \{\Omega_j, A_j, R_j^a, B_j, R_j^b \mid j \in \mathcal{A}\},$$

$$view_{\mathcal{A},received} = \{EA_k, EB_k, A_{ij}, R_{ij}^a, R_{ij}^{ab}, C_{ij}, R_{ij}^c,$$

$$EA_{ik}, EAB_{ik}, EC_{ik} \mid i \in \mathcal{H}, j \in \mathcal{A}, k = 0, \dots, t\},$$

where Ω_j is a private random tape of player j.

According to Lemma 1, for any \mathcal{A} of size no more than t, $view_{\mathcal{A},private}$ is independent of (A,B). Therefore, in order to prove the statement, it is sufficient to show that $view_{\mathcal{A},received}$ can be perfectly simulated. Given p,q,g,h and $\{EA_k,EB_k\mid k=0,\ldots,t\}$, execute the following steps for all $i\in\mathcal{H}$:

SIM-1: For $j \in \mathcal{A}$, choose A_{ij} and R_{ij}^a randomly from Z_q . Then compute EA_{ik} for k = 1, ..., t by solving the system of equations $g^{A_{ij}} h^{R_{ij}^a} = VA_i \prod_{k=1}^t EA_{ik}^{j^k}$ for $j \in \mathcal{A}$. (Note that this step does not compute discrete logarithms.)

SIM-2: For $j \in \mathcal{A}$, choose R_{ij}^{ab} randomly from Z_q . Also choose EAB_{i0} from G_q . Then, compute EAB_{ik} for k = 1, ..., t by solving $VB_i^{A_{ij}}h^{R_{ij}^{ab}} = \prod_{k=0}^{t} EAB_{ik}^{j^k}$ for $j \in \mathcal{A}$.

SIM-3: For $j \in \mathcal{A}$, choose C_{ij} , R_{ij}^c randomly from Z_q . Then, compute EC_{ik} for $k = 1, \ldots, t$ by solving $g^{C_{ij}} h^{R_{ij}^c} = EAB_{i0} \prod_{k=1}^t EC_{ik}^{j^k}$ for $j \in \mathcal{A}$.

As a result, A_{ij} , R_{ij}^a , R_{ij}^{ab} , C_{ij} , R_{ij}^c , EA_{ik} , EAB_{ik} , $EC_{ik} \mid i \in \mathcal{H}, j \in \mathcal{A}, k = 0, \ldots, t$ is obtained.

Let us examine the distribution of each variable. As A_{ij} , R^a_{ij} are randomly chosen in **SIM-1**, commitments EA_{ik} for $k=1,\ldots,t$ distribute uniformly over the space defined by relation $g^{A_{ij}}h^{R^a_{ij}}=VA_i\prod_{k=1}^t EA_{ik}^{j^k}$ for $j\in\mathcal{A}$, which is exactly the same relation that those shares and commitments satisfy in a real protocol run. Observe that A_{ij} , R^a_{ij} for $j\in\mathcal{A}$ randomly distribute over Z_q^{2t} in a real protocol run. Thus, A_{ij} , R^a_{ij} for $j\in\mathcal{A}$ and EA_{ik} for $k=1,\ldots,t$ have the same distribution as those in a real protocol run. A similar observation is possible for the remaining variables in $view_{\mathcal{A}.received}$.

Next consider a view in the disqualification protocol. Two cases arise: player j in \mathcal{A} challenges player i in \mathcal{H} , and the reverse case. In the former case, challenged player i in \mathcal{H} just broadcasts shares and commitments that have been already in the view of the adversary. In the latter case, every player i in \mathcal{H} just returns an answer based on the results of verification predicates in **DM-2**, which are

computed only with shares and commitments sent from the challenged player. Therefore, executing the disqualification protocol contributes nothing to $view_{\mathcal{A}}$. Secrecy in the share recovery protocol holds according to Theorem 5 of [19]. \square

Lemma 4 (Robustness). Let \mathcal{A} be a set of corrupted players. If $|\mathcal{A}| \leq t$ and $|\mathcal{P}| \geq 2t + 1$, every player i in $\mathcal{P} \setminus \mathcal{A}$ obtains correct share (C_i, R_i^c) and commitments EC_k for $k = 0, \ldots, t$ as a result of the above protocol.

Proof. Let $\mathcal{H} := \mathcal{P} \setminus \mathcal{A}$. Clearly, player i in \mathcal{H} will not be disqualified in the disqualification protocol because they are in the majority. According to Lemma 1, any inconsistency among shares and corresponding commitments is detected with overwhelming probability. Since the sender of the inconsistent shares is requested to broadcast the correct ones, what he can do is to broadcast the correct ones, the incorrect ones, or halt. Broadcasting incorrect shares or halting results in being disqualified immediately. Observe that share recovery protocol can be completed if there are at least t+1 honest players. So even if all corrupted players are disqualified at **DM-2**, the remaining players can complete share recovery protocol and obtain correct shares and commitments that should have sent from the disqualified player(s). Thus, there are at least t+1 players who have at least 2t+1 correct shares and commitments, which is sufficient to complete **DM-3** to obtain correct (C_j, R_i^c) and EC_k .

The above argument is correct if it is infeasible for player i to compute $\log_h VB_i$. Observe that

$$\log_h VB_i = \log_h g^{B_i} h^{R_i^b}$$
$$= B_i \log_h g + R_i^b.$$

Since we assume $\log_h g$ is not known to any player, player i could compute $\log_h VB_i$ only if $B_i = 0$. However, as B is assumed to be honestly shared, $B_i = 0$ happens with probability 1/q. Thus, the chance that player i distributes inconsistent shares remain negligible.

5 Extension

In the previous section we made the assumption that B is shared by an honest dealer so that $B_i = 0$ happens rarely. However, B is typically generated randomly by players in threshold cryptography. If RandVSS is used for this purpose, adversarial player i can control B_i so as to make it zero by distributing his choices after receiving shares from all other players.

By adding one more PedVSS to the multiplication protocol, we can deal with the case of $B_i = 0$. Let h_0 be a randomly chosen member of G_q . The idea is to let each player i share A_i with bases $[VB_i, h_0] = [g^{B_i} h^{R_i^b}, h_0]$ instead of $[VB_i, h]$. Accordingly, even if $B_i = 0$, player i can not cheat because he does not know $\log_{h_0} VB_i$ (= $R_i^b \log_{h_0} h$) unless $R_i^b = 0$. If both B_i and R_i^b equal 0, then other

players can detect it by finding $VB_i = 1$. The following is a brief description of the modified steps which correspond to **DM-1** and **DM-2**.

DM'-1: Each player i executes the following VSS-s.

$$PedVSS(A_i, R_i^a)[g, h] \xrightarrow{f_1, d_1} (A_{ij}, R_{ij}^a)(\langle VA_i \rangle, EA_{i1}, \dots, EA_{it})$$

$$PedVSS(A_i, R_i^{ab})[VB_i, h_0] \xrightarrow{f_1, d_2} (\langle A_{ij} \rangle, R_{ij}^{ab})(EAB_{i0}, \dots, EAB_{it})$$

$$PedVSS(A_i \cdot B_i, R_i^b \cdot A_i, R_i^{ab})[g, h, h_0] \xrightarrow{f_2, d_3, d_4} (C_{ij}, T1_{ij}, T2_{ij})$$

$$(\langle EAB_{i0} \rangle, ET_{i1}, \dots, ET_{it}),$$

$$PedVSS(A_i \cdot B_i, T3_i)[g, h] \xrightarrow{f_2, d_5} (\langle C_{ij} \rangle, T3_{ij})(EC_{i0}, \dots, EC_{it})$$

 R_i^{ab} and $T3_i$ are chosen randomly from Z_q . Intuitively, the first and second VSS commit $A_i \cdot B_i$ to EAB_{i0} . The third VSS distributes $A_i \cdot B_i$ committed to EAB_{i0} . The last VSS transforms the bases of commitment of $A_i \cdot B_i$ from $[g,h,h_0]$ to [g,h]. (The last VSS can be omitted depending on application. See the protocols in the next section for instance.)

DM'-2: Each player j verifies

$$VB_{i} \neq 1,$$

$$g^{A_{ij}} h^{R_{ij}^{a}} = VA_{i} \prod_{k=1}^{t} EA_{ik}^{j^{k}},$$

$$VB_{i}^{A_{ij}} h_{0}^{R_{ij}^{ab}} = \prod_{k=0}^{t} EAB_{ik}^{j^{k}},$$

$$g^{C_{ij}} h^{T1_{ij}} h_{0}^{T2_{ij}} = EAB_{i0} \prod_{k=1}^{t} ET_{ik}^{j^{k}},$$

$$g^{C_{ij}} h^{T3_{ij}} = \prod_{k=0}^{t} EC_{ik}^{j^{k}},$$

for all $i \in \mathcal{P}$.

Shares for player j can be computed in the same way as shown in **DM-3** by replacing R_{ij}^c with $T3_{ij}$. This extended multiplication protocol retains the all properties stated in the previous section except for the case where honest player i is disqualified because of $VB_i = 1$, though it happens with negligible probability as long as adversary is polynomially bounded. In the case of executing RandVSS for generating B in the presence of infinitely powerful adversaries, honest players must not be disqualified to maintain information theoretic secrecy. So if $VB_i = 1$ happens for player i, it declares so and invokes another PedVSS that is combined to the result of RandVSS so that $VB_i \neq 1$ holds for all $j \in \mathcal{P}$.

6 Application

Robust threshold encryption is a useful tool for constructing applications like secret ballot electronic voting schemes where a single trusted party is not preferred. The first robust threshold cryptosystem provably secure against adaptive chosen ciphertext attack was presented by Shoup and Gennaro in [25], where the security was proved in the random oracle model. The Cramer-Shoup cryptosystem [9] is a provably secure (non-threshold) scheme whose security can be proved only by using a standard cryptographic assumption, i.e., intractability of the decision Diffie-Hellman problem.

In [4], Canetti and Goldwasser introduced a robust threshold version of the Cramer-Shoup cryptosystem which requires four-move ZKP to tolerate a minority of corrupt players. Here we present another construction that enjoys lower round complexity and optimal resiliency. In our scenario, unlike previous one by Canetti et al., players (decryption servers) are assumed to use the decryption result themselves. Accordingly, it is the player who has to be convinced of the correctness of the result. Such a scenario will also be applicable to quorum controlled systems.

6.1 Cramer-Shoup Cryptosystem

Let g_1 and g_2 be independently chosen elements in G_q . A decryption key is 6-tuple 1 $(x_1,x_2,y_1,y_2,z_1,z_2) \in_R Z_q^6$, and the corresponding encryption key is triple (X,Y,Z) where $X=g_1^{x_1}\,g_2^{x_2},Y=g_1^{y_1}\,g_2^{y_2}$ and $Z=g_1^{z_1}\,g_2^{z_2}$. An encryption of a message, $M\in G_q$, is (u_1,u_2,v,m) that satisfies

$$u_1 = g_1^r, \ u_2 = g_2^r, \ m = M \cdot Z^r, \ c := Hash(u_1, u_2, m), \ v = X^r Y^{cr}$$

for $r \in_R Z_q$, where $Hash : \{0,1\}^* \to Z_q$ is a corrision-intractable hash function. To decrypt, compute

$$V := u_1^{x_1 + c \, y_1} u_2^{x_2 + c \, y_2}$$

and verify if V=v. If it is successful, the message is recovered by computing $M=m/u_1^{z_1}\,u_2^{z_2}$. Otherwise the encrypted message is rejected.

The security proof against adaptive chosen ciphertext attack in [9] utilizes the property that the decryption process leaks no information about each of the secret keys except for their linear relation such as $\log_{g_1} X = x_1 + x_2 \log_{g_1} g_2$.

A problem with the threshold scheme is that V must not be revealed to corrupt players if $V \neq v$. This means that players have to be convinced of $V \neq v$ without seeing V itself.

The original scheme presented in [9] only uses z_1 for efficiency. Here we use z_1 and z_2 for the sake of key generation, however, it never influences the security. Indeed, [9] proves the security using a model that uses z_1 and z_2 .

6.2 A Threshold Version

The underlying idea is to replace V with V' such as

$$V' := (V v^{-1})^w$$

where w is a one-time random number chosen from Z_q^* . Observe that V'=1 holds if and only if V=1 holds. Furthermore, for every $V' \in G_q$ and for every $V \in G_q$, there exists $w \in Z_q^*$ that satisfies $V' = (V \ v^{-1})^w$. Therefore, revealing V' does not leak any information about V in an information theoretic sense except when V'=1.

For efficiency, decryption is combined with the verification process by replacing the above V' with \tilde{V} such that

$$\begin{split} \tilde{V} &:= V'/u_1^{z_1} \; u_2^{z_2} \\ &= u_1^{\hat{x}_1 + c \; \hat{y}_1 - z_1} u_2^{\hat{x}_2 + c \; \hat{y}_2 - z_2} v^{-w}, \end{split}$$

where $\hat{x}_1 = w \, x_1 \mod q$, $\hat{x}_2 = w \, x_2 \mod q$, $\hat{y}_1 = w \, y_1 \mod q$ and $\hat{y}_2 = w \, y_2 \mod q$. Each player computes a share of \tilde{V} . This idea was introduced in [4].

For this to be done, players generate a random factor w for each ciphertext and multiply them by private keys to get $\hat{x}_1 := w \, x_1 \mod q$ and so on. This is where our multiplication protocol is applied. This step takes 2-moves but can be done before receiving the ciphertext. The decryption protocol can then be performed non-interactively.

Key Generation: Players generate a secret key $(x_1, x_2, y_1, y_2, z_1, z_2)$ by using RandVSS as follows.

$$RandVSS([x_1], [x_2])[g_1, g_2] \rightarrow (x_{1i}, x_{2i})(X, EX_1, \dots, EX_t)$$

 $RandVSS([y_1], [y_2])[g_1, g_2] \rightarrow (y_{1i}, y_{2i})(Y, EY_1, \dots, EY_t)$
 $RandVSS([z_1], [z_2])[g_1, g_2] \rightarrow (z_{1i}, z_{2i})(Z, EZ_1, \dots, EZ_t)$

Each player computes verification commitments $VX_i = X\prod_{k=1}^t E{X_k}^{i^k}$ and

 $VY_i = Y \prod_{k=1}^t EY_k^{i^k}$ for all $i \in \mathcal{P}$, and checks if they do not equal 1. Each player i then shares (z_{1i}, z_{2i}) as

$$PedVSS(z_{1i}, z_{2i})[g_1, g_2] \xrightarrow{f_1, f_2} (z_{1ij}, z_{2ij})(\langle VZ_i \rangle, EZ_{i1}, \dots, EZ_{it})$$

to prepare for decryption where VZ_i be verification commitments for shares z_{1i}, z_{2i} such that $VZ_i = \prod_{k=0}^t EZ_k^{i^k}$.

Key Randomization: Before performing the decryption protocol, players need to generate a shared random number w and obtain shares of products wx_1, wx_2, wy_1 , and wy_2 . Intuitively, this process corresponds to randomizing the private keys. In the following, g_3 is a random element of G_q whose indices for g_1 and g_2 are unknown.

[Protocol: KEY RANDOMIZATION]

KR-1 The players perform

$$RandVSS([w], [w'])[g_1, g_2] \rightarrow (w_i, w'_i)(EW_0, \dots, EW_t).$$

Let VW_i denote the verification commitment for w_i and w'_i .

KR-2 Each player P_i performs the following VSS-s with t-degree random polynomials $f_3, \ldots, f_7, d_1, \ldots, d_5$ whose free terms are chosen properly as indicated in left parentheses. $\gamma_{1i}, \ldots, \gamma_{4i}$ are randomly chosen from Z_q .

$$PedVSS(w_{i},w'_{i})[g_{1},g_{2}] \xrightarrow{f_{3},d_{1}} (w_{ij},w'_{ij})(\langle VW_{i}\rangle,EW_{i1},\cdots)$$

$$PedVSS(w_{i},\gamma_{1i})[VX_{i},g_{3}] \xrightarrow{f_{3},d_{2}} (\langle w_{ij}\rangle,\gamma_{1ij})(EWX_{i0},\cdots)$$

$$PedVSS(w_{i}x_{1i},w_{i}x_{2i},\gamma_{2i})[g_{1},g_{2},g_{3}] \xrightarrow{f_{4},f_{5},d_{3}} (\tilde{x}_{1ij},\tilde{x}_{2ij},\gamma_{2ij})$$

$$(\langle EWX_{i0}\rangle,E\tilde{X}_{i1},\cdots)$$

$$PedVSS(w_{i},\gamma_{3i})[VY_{i},g_{3}] \xrightarrow{f_{3},d_{4}} (\langle w_{ij}\rangle,\gamma_{3ij})(EWY_{i0},\cdots)$$

$$PedVSS(w_{i}y_{1i},w_{i}y_{2i},\gamma_{4i})[g_{1},g_{2},g_{3}] \xrightarrow{f_{6},f_{7},d_{5}} (\tilde{y}_{1ij},\tilde{y}_{2ij},\gamma_{4ij})$$

$$(\langle EWY_{i0}\rangle,E\tilde{Y}_{i1},\cdots)$$

KR-3 Each player j verifies everything in a way similar to that used in **DM-2**. Player j then computes its share of the randomized private keys as

$$(\hat{x}_{1j}, \hat{x}_{2j}, \hat{\gamma}_{2j}) := \left(\sum_{i \in \mathcal{I}} \lambda_{i,\mathcal{I}} \tilde{x}_{1ij}, \sum_{i \in \mathcal{I}} \lambda_{i,\mathcal{I}} \tilde{x}_{2ij}, \sum_{i \in \mathcal{I}} \lambda_{i,\mathcal{I}} \gamma_{2ij}\right), \text{ and}$$

$$(\hat{y}_{1j}, \hat{y}_{2j}, \hat{\gamma}_{4j}) := \left(\sum_{i \in \mathcal{I}} \lambda_{i,\mathcal{I}} \tilde{y}_{1ij}, \sum_{i \in \mathcal{I}} \lambda_{i,\mathcal{I}} \tilde{y}_{2ij}, \sum_{i \in \mathcal{I}} \lambda_{i,\mathcal{I}} \gamma_{4ij}\right)$$

in modulo q, where $\mathcal I$ is a set of qualified players. Player j also computes verification commitments

$$\begin{split} V\hat{X}_i &:= \prod_{k=0}^t \left(\prod_{\ell \in \mathcal{I}} E\hat{X}_{\ell k}^{\lambda_{\ell,\mathcal{I}}}\right)^{i^k}, \text{and} \\ V\,\hat{Y}_i &:= \prod_{k=0}^t \left(\prod_{\ell \in \mathcal{I}} E\,\hat{Y}_{\ell k}^{\lambda_{\ell,\mathcal{I}}}\right)^{i^k} \end{split}$$

for all $i\in\mathcal{I}$ to prepare for subsequent use. (Note that $V\hat{X}_i=g_1^{\hat{x}_{1i}}\,g_2^{\hat{x}_{2i}}\,g_3^{\gamma_{2i}}$ and $V\hat{Y}_i=g_1^{\hat{y}_{1i}}\,g_2^{\hat{y}_{2i}}\,g_3^{\gamma_{4i}}$.) Player j stores the shares and verification commitments.

[End]

Decryption:

[Protocol: Robust Threshold Decryption]

RD-1 Each player i broadcasts

$$\tilde{V}_i := u_1^{\hat{x}_{1i} + c \, \hat{y}_{1i} - z_{1i}} u_2^{\hat{x}_{2i} + c \, \hat{y}_{2i} - z_{2i}} v^{-w_i}$$

where $c := Hash(u_1, u_2, m) \mod q$. Player *i* then performs the following four VSS-s. Here, *t*-degree polynomials f_8, \ldots, f_{11} and d_6, d_7 are randomly chosen so that they have proper free terms as indicated in the parentheses in the right side.

$$PedVSS(\hat{x}_{1i}, \hat{x}_{2i}, \hat{\gamma}_{2i})[g_1, g_2, g_3] \xrightarrow{f_8, f_9, d_6} (\hat{x}_{1ij}, \hat{x}_{2ij}, \hat{\gamma}_{2ij})(\langle V \hat{X}_i \rangle, \ldots)$$

$$PedVSS(\hat{y}_{1i}, \hat{y}_{2i}, \hat{\gamma}_{4i})[g_1, g_2, g_3] \xrightarrow{f_{10}, f_{11}, d_7} (\hat{y}_{1ij}, \hat{y}_{2ij}, \hat{\gamma}_{4ij})(\langle V \hat{Y}_i \rangle, \ldots)$$

and

$$PedVSS(\hat{x}_{1i} + c\,\hat{y}_{1i} - z_{1i}, \,\hat{x}_{2i} + c\,\hat{y}_{2i} - z_{2i}, \, -w_i)[u_1, u_2, v] \xrightarrow{f_{12}, f_{13}, f_{14}} (\langle \hat{x}_{1ij} + c\,\hat{y}_{1ij} - z_{1ij} \rangle, \langle \hat{x}_{2ij} + c\,\hat{y}_{2ij} - z_{2ij} \rangle, \langle -w_{ij} \rangle)(\langle \tilde{V}_i \rangle, E\,\tilde{V}_{i1}, \cdots, E\,\tilde{V}_{it}),$$

where $f_{12} = f_8 + c f_{10} - f_1$, $f_{13} = f_9 + c f_{11} - f_2$, and $f_{14} = -f_3$.

RD-2 Each player j verifies

$$\begin{split} g_1^{\hat{x}_{1ij}} \, g_2^{\hat{x}_{2ij}} \, g_3^{\hat{\gamma}_{2ij}} &= \, V \hat{X}_i \prod_{k=1}^t E \hat{X}_{ik}^{j^k}, \\ g_1^{\hat{y}_{1ij}} \, g_2^{\hat{y}_{2ij}} \, g_3^{\hat{\gamma}_{4ij}} &= \, V \, \hat{Y}_i \prod_{k=1}^t E \, \hat{Y}_{ik}^{j^k}, \text{ and} \\ u_1^{\hat{x}_{1ij} + c \, \hat{y}_{1ij} - z_{1ij}} \, u_2^{\hat{x}_{2ij} + c \, \hat{y}_{2ij} - z_{2ij}} \, v^{-w_{ij}} &= \, \tilde{V}_i \prod_{k=1}^t E \, \tilde{V}_{ik}^{j^k} \end{split}$$

for all i received. Disqualification will be done in the similar way as before. Player j finally obtains plaintext as

$$M = m / \prod_{i \in \mathcal{Q}} \tilde{V}_i^{\lambda_{i,\mathcal{Q}}}$$

where Q is a set of more than t+1 qualified players.

[End]

6.3 Security Analysis

Our definition of security against an adaptive chosen ciphertext attack refers to [9]. Similarly, a precise security model for the multi-party setting can be seen in [25]. In that model, there are adversaries inside the decryption oracle

who may leak internal information to an attacker. However, because individual private keys and verification result, i.e., V are information theoretically secure during the protocols, the joint view of the attacker and the insiders does not differ from that of the original scheme except for the following:

- Since RandVSS produces $w \in_U Z_q$, instead of Z_q^* , w = 0 happens with probability 1/q. In this case, $\tilde{V} = 1$ holds regardless of V. Accordingly, an incorrect message is accepted with probability 1/q.
- An adversary has a chance to get \tilde{V} to be 1 for incorrect messages by adjusting corrupted player's share \tilde{V}_i after seeing shares sent from uncorrupted players. However, this forces the corrupt player i to share \tilde{V}_i without knowing its witnesses in **RD-1**. Hence the success probability is 1/q.
- Honest player j can be disqualified with probability 2/q, because of $VX_j=1$ or $VY_j=1$.

All of those increases in the probability of accepting incorrect messages are negligible in |q| if the number of adversaries is limited to polynomial in |q|. Thus, the players reject all invalid ciphertexts except with negligible probability. The rest of the protocol is the same as the proof of security for the original scheme.

7 Conclusion

We have shown a non-interactive distributed multiplication protocol in standard cryptographic setting with a private network and broadcast channels. Non-interactive means that players need to use the private network and broadcast channel only once to send data without the need to synchronize with other players. The protocol withstands less than n/2 corrupt players and uses no zero-knowledge interactive proofs. Compared to the previous four-move scheme that relies on zero-knowledge proofs, it increases computation and communication complexity according to the factor of threshold t.

As an application of our protocol, we constructed a threshold version of the Cramer-Shoup cryptosystem which works non-interactively with precomputation for randomizing private keys. If the key randomization protocol is combined to the decryption protocol, it requires four data moves. Although this form still enjoys less round complexity than that of the previous construction (including precomputation steps), it raises an open problem: Show an optimally resilient and non-interactive protocol that generates a random number and multiplies it by a previously shared secret.

Acknowledgments

The author wishes to thank Rosario Gennaro for his invaluable suggestions. Comments from Tatsuaki Okamoto and Eiichiro Fujisaki on an early version of this paper are also appreciated.

References

- 1. D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4:75–122, 1991.
- M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncyptographic fault-tolerant distributed computation. In *Proceedings of the 20th* annual ACM Symposium on the Theory of Computing, pages 1–10, 1988.
- 3. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In B. S. Kaliski Jr., editor, *Advances in Cryptology CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 425–439. Springer-Verlag, 1997.
- 4. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Jacques Stern, editor, *Advances in Cryptology EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 90–106. Springer-Verlag, 1999.
- M. Cerecedo, T. Matsumoto, and H. Imai. Efficient and secure multiparty generation of digital signatures based on discrete logarithms. *IEICE Transaction of Fundamentals of electronic Communications and Computer Science*, E76-A(4):532–545, April 1993.
- D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In Proceedings of the 20th annual ACM Symposium on the Theory of Computing, pages 11–19, 1988.
- R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, Advances in Cryptology — EUROCRYPT '99, volume 1592 of Lecture Notes in Computer Science, pages 311–326. Springer-Verlag, 1999.
- R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In W. Fumy, editor, Advances in Cryptology — EUROCRYPT '97, volume 1233 of Lecture Notes in Computer Science, pages 103– 118. Springer-Verlag, 1997.
- 9. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.
- Y. G. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, Advances in Cryptology — CRYPTO '89, volume 435 of Lecture Notes in Computer Science, pages 307–315. Springer-Verlag, 1990.
- 11. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th IEEE Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.
- P. Feldman and S. Micali. Optimal algorithms for byzantine agreement. In Proceedings of the 20th ACM Symposium on the Theory of Computing, pages 148–161, 1988.
- A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, Advances in Cryptology — CRYPTO '86, volume 263 of Lecture Notes in Computer Science, pages 186–199. Springer-Verlag, 1986.
- Y. Frankel, P. D. MacKenzie, and M. Yung. Robust efficient distributed RSAkey generation. In *Proceedings of the 30th annual ACM Symposium on Theory of Computing*, pages 663–672, New York City, 1998.

- R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In U. Maurer, editor, Advances in Cryptology EUROCRYPT '96, volume 1070 of Lecture Notes in Computer Science, pages 354–371. Springer-Verlag, 1996.
- R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In Jacques Stern, editor, Advances in Cryptology EUROCRYPT '99, volume 1592 of Lecture Notes in Computer Science, pages 295–310. Springer-Verlag, 1999.
- R. Gennaro, M. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In 17th ACM Symposium on Principles of Distributed Computing, 1998.
- 18. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the 27th IEEE Annual Symposium on Foundations of Computer Science*, pages 174–187, 1986.
- A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In D. Coppersmith, editor, Advances in Cryptology — CRYPTO '95, volume 963 of Lecture Notes in Computer Science, pages 339–352. Springer-Verlag, 1995.
- M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 2(27):228–234, 1980.
- 21. T. P. Pedersen. A threshold cryptosystem without a trusted party. In D. W. Davies, editor, Advances in Cryptology EUROCRYPT '91, volume 547 of Lecture Notes in Computer Science, pages 522–526. Springer-Verlag, 1991.
- T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, Advances in Cryptology — CRYPTO '91, volume 576 of Lecture Notes in Computer Science, pages 129–140. Springer-Verlag, 1992.
- 23. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21st annual ACM Symposium on the Theory of Computing*, pages 73–85, 1989.
- 24. A. Shamir. How to share a secret. Communications of the ACM, 22:612–613, 1979.
- V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In K. Nyberg, editor, Advances in Cryptology — EUROCRYPT '98, Lecture Notes in Computer Science, pages 1–16. Springer-Verlag, 1998.

A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting

Berry Schoenmakers

Department of Mathematics and Computing Science Eindhoven University of Technology P.O. Box 513, 5600 MB Eindhoven, The Netherlands berry@win.tue.nl

Abstract. A publicly verifiable secret sharing (PVSS) scheme is a verifiable secret sharing scheme with the property that the validity of the shares distributed by the dealer can be verified by any party; hence verification is not limited to the respective participants receiving the shares. We present a new construction for PVSS schemes, which compared to previous solutions by Stadler and later by Fujisaki and Okamoto, achieves improvements both in efficiency and in the type of intractability assumptions. The running time is O(nk), where k is a security parameter, and n is the number of participants, hence essentially optimal. The intractability assumptions are the standard Diffie-Hellman assumption and its decisional variant. We present several applications of our PVSS scheme, among which is a new type of universally verifiable election scheme based on PVSS. The election scheme becomes quite practical and combines several advantages of related electronic voting schemes, which makes it of interest in its own right.

1 Introduction

Secret sharing and its many variations form an important primitive in cryptography. The basic model for secret sharing distinguishes at least two protocols: (i) a distribution protocol in which the secret is distributed by a dealer among the participants, and (ii) a reconstruction protocol in which the secret is recovered by pooling the shares of a qualified subset of the participants. Basic schemes (e.g., [Sha79,Bla79] for threshold secret sharing) solve the problem for the case that all players in the scheme are honest.

In verifiable secret sharing (VSS) the object is to resist malicious players, such as

- (i) a dealer sending incorrect shares to some or all of the participants, and
- (ii) participants submitting incorrect shares during the reconstruction protocol,

cf. [CGMA85]. In publicly verifiable secret sharing (PVSS), as introduced by Stadler [Sta96], it is an explicit goal that not just the participants can verify their own shares, but that anybody can verify that the participants received correct shares. Hence, it is explicitly required that (i) can be verified publicly. (As noted

in [Sta96], the VSS scheme of [CGMA85] already achieved this property, but later VSS schemes weren't designed to be publicly verifiable.) Problem (ii) is usually dealt with implicitly though. In the schemes of [Fel87,Ped92b,Sta96,FO98] it suffices that the participants simply release their shares. Subsequently the released shares may be verified by anybody against the output of the distribution protocol.

Our PVSS schemes show that such an approach is not sufficient as a general model for PVSS. As an extension to the reconstruction protocol we will therefore require that the participants not only release their shares but also that they provide a proof of correctness for each share released (see Section 2).

For PVSS schemes it is natural to accept that the secret is computationally hidden. An information-theoretic hiding VSS scheme such as [Ped92a] requires the availability of *private* channels from the dealer to each of the participants individually. However, communication over the private channels is clearly not publicly verifiable.

Given that a PVSS scheme is computationally hiding, the question is what the exact number-theoretic assumptions are. For our new scheme, which works for any group for which the discrete log problem is intractable, we will relate the security to the Diffie-Hellman assumption and its decisional variant. These assumptions are common for encryption schemes in a discrete log setting. In fact, there is a direct connection with the security of the ElGamal cryptosystem, as, for instance, the semantic security of ElGamal encryption is equivalent to the Diffie-Hellman decision problem. So, in a sense, this type of assumption is the weakest one can hope for.

In contrast, the schemes of [Sta96,FO98] rely on special number-theoretic settings and intractability assumptions. The discrete log scheme of [Sta96] requires a special assumption involving "double discrete logarithms". Briefly, the idea is to consider expressions of the form $y = g^{(h^x)}$, where g is a generator of a group of order p, say, and h is a fixed element of high order in \mathbb{Z}_p^* . The "double discrete logarithm" assumption states that given y it is hard to find x, which is a non-standard assumption.

We observe that such a setting excludes the use of elliptic curves. Let us call the group generated by g the base group, and the group generated by h the exponent group. Now, since the notion of double logarithms relies on the fact that h^x can be interpreted as an integer modulo p (since $h^x \in \mathbb{Z}_p^*$), the exponent group can not simply be replaced by a group based on elliptic curves. The security of Stadler's scheme requires that the discrete problem for the exponent group is hard as well, hence p must be 1024 bits, say. We conclude that although the base group can be based on elliptic curves, its order p would be rather high (normally, elliptic curve cryptosystems attain their competitiveness by requiring the order to be of size 160 bits, say).

The scheme of [FO98] relies on what they call the "modified RSA assumption," which says that inverting the RSA function is still hard, even if the public exponent e may be chosen freely. Although this assumption is not known to be actually stronger than the RSA assumption, it potentially is. Furthermore,

the number-theoretic setting for the primitives used (e.g., secure commitments) are computationally expensive and require a non-trivial set-up for the system, using zero-knowledge proofs to show that set-up is correct. Stadler [Sta96] also considers a PVSS scheme for sharing e-th roots, which at least depends on the RSA assumption and variants of the Diffie-Hellman assumption for composite moduli.

Summarizing, our PVSS construction will be much simpler than the above approaches. We only need techniques that work in any group for which the discrete log problem is intractable. The protocols consist of a few steps only, relying on simple primitives. The performance is not only asymptotically optimal, but also good in practice. And, finally, the PVSS scheme can readily be used to extend all kinds of applications without strengthening the security assumptions or essentially increasing the computational cost of the resulting scheme. This is essential in order for a primitive to be used in a modular fashion.

1.1 Overview

In Section 2 we will describe the characteristics of PVSS schemes. The model we propose follows [Sta96] with some refinements, and we consider what it means for a PVSS scheme to be homomorphic.

In Section 3 we then present our main construction of a simple PVSS scheme for sharing secrets chosen uniformly at random from a large set. The complexity of the PVSS scheme is linear in the security parameter k (and linear in the number of participants), which is essentially optimal. As a result our scheme is a factor of k more efficient than Stadler's discrete log scheme [Sta96], and it achieves the same asymptotic complexity as the scheme for e-th roots of [Sta96] and the PVSS scheme of [FO98]; as we will point out, however, the concrete running time for our scheme is significantly lower than for these schemes.

In Section 4 we consider extensions to the case that the secret is from a small set. Our PVSS schemes work for any group of prime order for which the discrete logarithm is hard. Its security relies on the standard Diffie-Hellman assumption and its decisional variant, which makes the PVSS scheme as secure as the ElGamal cryptosystem. We stress that, unlike [Sta96,FO98], we consider also whether a PVSS scheme leaks partial information on the secret. Similarly, we do not only consider PVSS for uniformly random secrets from a large set, but also for secrets from a small set, which need not be uniformly distributed either. This is of importance to applications such as electronic voting, in which the secret may consist of a single bit. (In Appendix A, we also show that the construction works for any access structure, as long as it admits a linear secret sharing scheme.)

In Section 5 we present a new approach to electronic voting based on PVSS. It turns out that the PVSS scheme exactly matches the discrete log setting and assumptions required for the remainder of the election scheme. The resulting scheme combines several of the advantages of previous election schemes, and fits conceptually between the schemes of [CFSY96] and [CGS97]. It achieves the same set of security properties as these schemes.

Finally, Section 6 contains several more applications of our PVSS schemes, including a generalization of Binding ElGamal [VT97] and other constructions related to software key escrow, such as [YY98]. We show how threshold variants for these schemes can be obtained easily.

2 Model for Non-interactive PVSS

In this section we first describe a model for non-interactive PVSS. We then discuss some aspects of the model, where we will also address the issue of homomorphic secret sharing for PVSS schemes.

We note that a distinctive feature of PVSS is that no private channels between the dealer and the participants are assumed. All communication is done over (authenticated) public channels using public key encryption. Consequently, the secret will only be hidden computationally.

In a PVSS scheme, a dealer D wishes to distribute shares of a secret value $s \in \Sigma$ among n participants P_1, \ldots, P_n . A monotone access structure describes which subsets of participants are qualified to recover the secret. For example, the access structure may be a (t, n)-threshold schemes, $1 \le t \le n$, which means that any subset of t or more participants will be able to recover the secret; any smaller subset will be unable to gain any information about the secret, unless a computational assumption is broken.

As a common structure for PVSS schemes we consider the following protocols. Note that initialization is done without any interaction between the dealer and the participants. In fact, participants may enter or leave the system *dynamically*; the only requirement is that a participant holds a registered public key.

Initialization All system parameters are generated as part of the initialization. Furthermore, each participant P_i registers a public key to be used with a public key encryption method E_i . The actual set of participants taking part in a run of the PVSS scheme must be a subset of the registered participants. We assume w.l.o.g. that participants P_1, \ldots, P_n are the actual participants in the run described below.

Distribution The protocol consists of two steps:

- 1. Distribution of the shares. The distribution of a secret $s \in \Sigma$ is performed by the dealer D. The dealer first generates the respective shares s_i for participant P_i , for $i=1,\ldots,n$. For each participant P_i the dealer publishes the encrypted share $E_i(s_i)$. The dealer also publishes a string $PROOF_D$ to show that each E_i encrypts a share s_i . Furthermore, the string $PROOF_D$ commits the dealer to the value of secret s, and it guarantees that the reconstruction protocol will result in the same value s.
- 2. Verification of the shares. Any party knowing the public keys for the encryption methods E_i may verify the shares. For each participant P_i a non-interactive verification algorithm can be run on $PROOF_D$ to verify that $E_i(s_i)$ is a correct encryption of a share for P_i . Since anyone may verify a share, it may be ruled out that a participant complains while it received a

correct share. In case one or more verifications fail, we therefore say that the dealer fails, and the protocol is aborted. (If some level of fault-tolerance is desired one may continue and think of it as a (t, n-c)-threshold scheme, where c is the number of verifications that failed.)

Reconstruction The protocol consists of two steps:

- 1. Decryption of the shares. The participants decrypt their shares s_i from $E_i(s_i)$. It is not required that all participants succeed in doing so, as long as a qualified set of participants is successful. These participants release s_i plus a string $PROOF_{P_i}$ that shows that the released share is correct.
- 2. Pooling the shares. The strings $PROOF_{P_i}$ are used to exclude the participants which are dishonest or fail to reproduce their share s_i correctly. Reconstruction of the secret s can be done from the shares of any qualified set of participants.

Compared to [Sta96], we have added the requirement for the reconstruction protocol that the participants must provide a proof of correct decryption of their shares. The proof is also non-interactive so that any party is able to sort out the correct shares and pool them together.

We have limited the description to non-interactive PVSS schemes by requiring that all PROOFs can be verified non-interactively. In fact, it is natural to reduce the amount of interaction between the players even more than for VSS schemes. Non-interactive VSS schemes, such as [Fel87,Ped92b], still include a stage in which participants file complaints if they received an incorrect share. Subsequently these complaints must be resolved to decide whether the distribution of the secret was successful. In non-interactive PVSS we have eliminated even this round of interaction: since any party can verify the output of the dealer, there is no need for the individual participants to check their own shares!

Homomorphic Secret Sharing The notion of homomorphic secret sharing is due to Benaloh [Ben87a], where its relevance to several applications of secret sharing is described, in particular electronic voting. Informally, homomorphic secret sharing is about combining shares of independent secrets in such a way that reconstruction from the combined shares results in a combined secret. In case of PVSS, there is an operation \oplus on the shares, and an operation \otimes on the encrypted shares such that for all participants

$$E_i(s_i) \otimes E_i(s'_i) = E_i(s_i \oplus s'_i).$$

Thus by decrypting the \otimes -combined encrypted shares, the recovered secret will be equal to $s \oplus s'$, assuming that the underlying secret sharing scheme is \oplus -homomorphic. In Section 5 we will present an electronic voting scheme that relies on a homomorphic PVSS scheme.

3 Special PVSS Scheme

We describe the construction for a (t, n)-threshold access structure, but it can be applied to any monotone access structure for which a linear secret sharing scheme exists (see Appendix A).

Let G_q denote a group of prime order q, such that computing discrete logarithms in this group is infeasible. Let g, G denote independently selected generators of G_q , hence no party knows the discrete log of g with respect to G. We solve the problem of efficiently sharing a random value from G_q . The dealer will achieve this by first selecting $s \in_R \mathbb{Z}_q$ and then distributing shares of the secret $S = G^s$. This approach allows us to keep the required proofs simple and efficient.

3.1 Protocols

We will use the protocol by Chaum and Pedersen [CP93] as a subprotocol to prove that $\log_{g_1} h_1 = \log_{g_2} h_2$, for generators $g_1, h_1, g_2, h_2 \in G_q$. We denote this protocol by $DLEQ(g_1, h_1, g_2, h_2)$, and it consists of the following steps, where the prover knows α such that $h_1 = g_1^{\alpha}$ and $h_2 = g_2^{\alpha}$:

- 1. The prover sends $a_1 = g_1^w$ and $a_2 = g_2^w$ to the verifier, with $w \in_R \mathbb{Z}_q$.
- 2. The verifier sends a random challenge $c \in_R \mathbb{Z}_q$ to the prover.
- 3. The prover responds with $r = w \alpha c \pmod{q}$.
- 4. The verifier checks that $a_1 = g_1^r h_1^c$ and $a_2 = g_2^r h_2^c$.

Initialization The group G_q and the generators g, G are selected using an appropriate public procedure. Participant P_i generates a private key $x_i \in_R \mathbb{Z}_q^*$ and registers $y_i = G^{x_i}$ as its public key.

Distribution The protocol consists of two steps:

1. Distribution of the shares. Suppose w.l.o.g. that the dealer wishes to distribute a secret among participants P_1, \ldots, P_n . The dealer picks a random polynomial p of degree at most t-1 with coefficients in \mathbb{Z}_q :

$$p(x) = \sum_{j=0}^{t-1} \alpha_j x^j,$$

and sets $s = \alpha_0$. The dealer keeps this polynomial secret but publishes the related commitments $C_j = g^{\alpha_j}$, for $0 \le j < t$. The dealer also publishes the encrypted shares $Y_i = y_i^{p(i)}$, for $1 \le i \le n$, using the public keys of the participants. Finally, let $X_i = \prod_{j=0}^{t-1} C_j^{ij}$. The dealer shows that the encrypted shares are consistent by producing a proof of knowledge of the unique p(i), $1 \le i \le n$, satisfying:

$$X_i = g^{p(i)}, \qquad Y_i = y_i^{p(i)}.$$

The non-interactive proof is the n-fold parallel composition of the protocols for $DLEQ(g, X_i, y_i, Y_i)$. Applying Fiat-Shamir's technique, the challenge c for the protocol is computed as a cryptographic hash of $X_i, Y_i, a_{1i}, a_{2i}, 1 \le i \le n$. The proof consists of the common challenge c and the n responses r_i .

2. Verification of the shares. The verifier computes $X_i = \prod_{j=0}^{t-1} C_j^{i^j}$ from the C_j values. Using $y_i, X_i, Y_i, r_i, 1 \le i \le n$ and c as input, the verifier computes a_{1i}, a_{2i} as

$$a_{1i} = g^{r_i} X_i^c, \qquad a_{2i} = y_i^{r_i} Y_i^c,$$

and checks that the hash of $X_i, Y_i, a_{1i}, a_{2i}, 1 \le i \le n$, matches c.

Reconstruction The protocol consists of two steps:

- 1. Decryption of the shares. Using its private key x_i , each participant finds the share $S_i = G^{p(i)}$ from Y_i by computing $S_i = Y_i^{1/x_i}$. They publish S_i plus a proof that the value S_i is a correct decryption of Y_i . To this end it suffices to prove knowledge of an α such that $y_i = G^{\alpha}$ and $Y_i = S_i^{\alpha}$, which is accomplished by the non-interactive version of the protocol $DLEQ(G, y_i, S_i, Y_i)$.
- 2. Pooling the shares. Suppose w.l.o.g. that participants P_i produce correct values for S_i , for i = 1, ..., t. The secret G^s is obtained by Lagrange interpolation:

$$\prod_{i=1}^{t} S_i^{\lambda_i} = \prod_{i=1}^{t} \left(G^{p(i)} \right)^{\lambda_i} = G^{\sum_{i=1}^{t} p(i)\lambda_i} = G^{p(0)} = G^s,$$

where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ is a Lagrange coefficient.

Note that the participants do not need nor learn the values of the exponents p(i). Only the related values $S_i = G^{p(i)}$ are required to complete the reconstruction of the secret value $S = G^s$. Also, note that participant P_i does not expose its private key x_i ; consequently participant P_i can use its key pair in several runs of the PVSS scheme. The type of encryption used for the shares has been optimized for performance; however, if desired, it is also possible to use standard ElGamal encryption instead.

Clearly, the scheme is homomorphic. For example, given the dealer's output for secrets G^{s_1} and G^{s_2} , the combined secret $G^{s_1+s_2}$ can be obtained by applying the reconstruction protocol to the combined encrypted shares $Y_{i1}Y_{i2}$. We will use this to construct an election scheme in Section 5.

3.2 Performance

The dealer only needs to post t+n elements of G_q (the numbers C_j and Y_i) plus n+1 number of size |q| (the responses r_i and the challenge c). The number of exponentiations throughout the protocol is correspondingly low, and all of these exponentiations are with relatively *small* exponents from \mathbb{Z}_q (|q| = 160 bits in practice).

Compared to Stadler's $O(k^2n)$ discrete log scheme, where k is a security parameter, we have reduced the work to O(kn), which is asymptotically optimal. Compared to the e-th root scheme of Stadler [Sta96] and the scheme of [FO98], which achieve the same asymptotic complexity as our construction, our scheme is much simpler. The construction of [FO98] uses a rather complicated proof to show that a share is correctly encrypted. For example, in its simplest form, that is, using RSA with public exponent 3, the scheme requires at least 17 secure commitments per participant, where each commitment requires a two-way or three-way exponentiation. Moreover, the exponents are of full-length (1024 bits in practice). Therefore, we estimate our scheme to be faster by a factor of 25 to 50. Similarly, but to a lesser extent, the primitive operations for Stadler's e-th root scheme, are more costly than for our scheme.

3.3 Security

We first consider the security of the share-encryptions. We observe that directly breaking the encryptions used in our PVSS scheme implies breaking the Diffie-Hellman assumption. This can be seen as follows. Breaking the encryption of the shares amounts to finding $G^{p(i)}$ given g, G, X_i, y_i, Y_i , for the group G_q . Writing $G = g^{\alpha}$, $X_i = g^{\beta}$, $y_i = g^{\gamma}$, breaking the encryption of the shares is equivalent to computing $g^{\alpha\beta}$, given g^{α} , g^{β} , g^{γ} , and $g^{\beta\gamma}$, for $\alpha, \beta, \gamma \in_R \mathbb{Z}_q$. Recalling that the Diffie-Hellman assumption states that it is infeasible to compute $g^{\alpha\beta}$, given g^{α} and g^{β} , we have the following lemma.

Lemma 1. Under the Diffie-Hellman assumption, it it infeasible to break the encryption of the shares.

Proof. Given $x = g^{\alpha}$ and $y = g^{\beta}$, we want to obtain $z = g^{\alpha\beta}$ by using an algorithm \mathcal{A} that breaks the encryption of the shares. Pick random α', β', γ , and feed $x^{\alpha'}, y^{\beta'}, g^{\gamma}, y^{\beta'\gamma}$ to \mathcal{A} . Since the input to \mathcal{A} is uniformly distributed, we then obtain $z' = g^{\alpha\alpha'\beta\beta'}$ with some success probability ϵ . By taking $z'^{1/(\alpha'\beta')} = g^{\alpha\beta}$, we are thus able to compute z with the same success probability ϵ .

A stronger result is that the secret is protected unless t or more participants cooperate. This is expressed by the following lemma.

Lemma 2. Suppose that t-1 participants pool their shares and obtain the secret. Then we can break the Diffie-Hellman assumption.

Proof. Let g^{α} and g^{β} be given, so we want to obtain $g^{\alpha\beta}$. We assume that α and β are random; if not, it is trivial to adapt the proof, as in the previous lemma. Suppose w.l.o.g. that participants P_1, \ldots, P_{t-1} are able to break the scheme. We will show how to set up the system such that this fact enables us to compute $g^{\alpha\beta}$.

We put $G = g^{\alpha}$ and $C_0 = g^{\beta}$, which implicitly defines p(0) as it is required that $C_0 = g^{p(0)}$. The points $p(1), \ldots, p(t-1)$ are chosen at random from \mathbb{Z}_q , which fixes polynomial p. This allows us to directly compute $X_i = g^{p(i)}$ and

 $Y_i = y_i^{p(i)}$, for i = 1, ..., t - 1. Since p(0) is only given implicitly, we cannot compute the points p(t), ..., p(n). It suffices, however, that we can compute $X_i = g^{p(i)}$ by Lagrange interpolation, which also yields the remaining C_j 's. We now deviate from the protocol by computing the public keys y_i of participants P_i , i = t, ..., n, as $y_i = g^{w_i}$ for random $w_i \in \mathbb{Z}_q$, and we set $Y_i = X_i^{w_i}$ such that $Y_i = y_i^{p(i)}$, as required.

The complete view for the system is now defined. It is consistent with the private view of participants P_1, \ldots, P_{t-1} , and comes from the right distribution. Now, suppose that they are able to compute the secret $G^{p(0)}$. Since $G = g^{\alpha}$ and $p(0) = \beta$, we are thus able to compute $g^{\alpha\beta}$. This contradicts the Diffie-Hellman assumption.

Note that we are assuming a static adversary. The above argument may be extended to the case where the static adversary is allowed to take part in the PVSS protocol K times, say, before breaking it. In that case we follow the protocol (hence we know the polynomials p) for the first K runs except that for participants P_t, \ldots, P_n we will set $S_i = G^{p(i)}$ directly instead of decrypting Y_i .

So far we have ignored the proofs that are required at several points in the protocol. However, in the random oracle model these proofs can easily be simulated. This leads to the following summary.

Theorem 1. Under the Diffie-Hellman assumption, the special PVSS scheme is secure in the random oracle model. That is, (i) the reconstruction protocol results in the secret distributed by the dealer for any qualified set of participants, (ii) any non-qualified set of participants is not able to recover the secret.

Proof. It follows from the soundness of the Chaum-Pedersen proof and the fact that the X_i 's are obtained from the C_j 's as $X_i = \prod_{j=0}^{t-1} C_j^{i^j}$ that the shares of the participants are consistent with the secret. It follows from Lemma 2 and the fact that the Chaum-Pedersen proof is honest-verifier zero-knowledge (hence the non-interactive version releases no information under the random oracle assumption) that no set of less than t participants can recover the secret.

Theorem 1 does not claim that the participants cannot get any partial information on the secret G^s . This stronger result holds under the assumption that ElGamal encryption is semantically secure, which is known to be equivalent to the Decision DH assumption. The latter assumption states that it is infeasible to determine whether a given triple is of the form $(g^{\alpha}, g^{\beta}, g^{\alpha\beta})$ or $(g^{\alpha}, g^{\beta}, g^{\delta})$, for random α, β, δ .

The above results are easily adapted to this case. For the equivalent of Lemma 1 we reason as follows. Suppose that an adversary is able to determine whether an encrypted share is equal to a given value g^{δ} or not. We then obtain a contradiction with the Decision DH assumption, closely following Lemma 1, by setting $G = g^{\alpha}$, $X_i = g^{\beta}$, and for random γ , setting $y_i = g^{\gamma}$ and $Y_i = (X_i)^{\gamma} = g^{\beta\gamma}$. Since the share is equal to $G^{\beta} = g^{\alpha\beta}$ it follows that we are able to distinguish $g^{\alpha\beta}$ from g^{δ} , if the adversary is able to distinguish the share from g^{δ} . The equivalent of Lemma 2 can be proved in a similar way. This leads to the following conclusion.

Theorem 2. Under the DDH assumption and the random oracle assumption, the special PVSS scheme is secure. That is, (i) the reconstruction protocol results in the secret distributed by the dealer for any qualified set of participants, (ii) any non-qualified set of participants is not able to recover any (partial) information on the secret.

4 General PVSS Schemes

The special PVSS scheme solves the basic problem of sharing a random secret from G_q . In this section we show how to extend this to any type of given secret $\sigma \in \Sigma$, where $2 \leq |\Sigma| \leq q$. Hence, a small set $|\Sigma| = 2$ is allowed, and it is not required that σ is uniformly distributed over Σ . We describe two methods.

For the first method, the general procedure is to let the dealer first run the distribution protocol for a random value $s \in \mathbb{Z}_q$, and then publish $U = \sigma \oplus \mathcal{H}(G^s)$, where \mathcal{H} is an appropriate cryptographic hash function. The reconstruction protocol will yield G^s , from which we obtain $\sigma = U \oplus \mathcal{H}(G^s)$. See Section 6.1 for a similar technique. More generally, U may be viewed as an encryption of σ under the key G^s .

A second, more specific, method for the case that $\Sigma \subseteq G_q$ works as follows. The dealer runs the distribution protocol for s and also publishes $U = \sigma G^s$. Upon reconstruction of G^s , this will yield $\sigma = U/G^s$. Extending Theorem 2, security is maintained under the Decision DH assumption, even if Σ is a small set. (Recall that $C_0 = g^s$ is part of the output of the distribution protocol of our special PVSS scheme. Hence together with U this constitutes an ElGamal encryption of the form $(g^s, G^s\sigma)$.) This method also allows us to share bit strings without using a hash function as in the first method, e.g., if $\Sigma = \{1, \ldots, G^{2^u-1}\}$ we may share u bits at once, as long as $\log_G \sigma$ can be computed efficiently for $\sigma \in \Sigma$.

For the above methods, it does not necessarily follow that for a given value of U the reconstructed σ will be an element of Σ . However, it does follow for the first method in the common case that $\Sigma = \{0,1\}^u$ and the range of \mathcal{H} is also equal to $\{0,1\}^u$, and also for the second method in case $\Sigma = G_q$. In other cases, an additional proof of knowledge may be required to show that indeed $\sigma \in \Sigma$ (e.g., see the next section), or, depending on the application, it may be sufficient to discard σ 's outside Σ (or replace it by a default value in Σ).

5 Electronic Voting

In this section we briefly show how to construct a universally verifiable secretballot election scheme using PVSS as a basic tool. We show that by using our PVSS scheme we get a simple and efficient election scheme. This is not the case for the PVSS schemes of [Sta96] and [FO98].

We follow the model for universally verifiable elections as introduced by Benaloh *et al.* [CF85,BY86,Ben87b], which assumes the availability of a so-called

bulletin board, to which all of the players in the scheme will post their messages. The players comprise a set of tallying authorities (talliers) A_1, \ldots, A_n , a set of voters V_1, \ldots, V_m , and a set of passive observers. These sets need not be disjoint. For example, in small-scale elections (e.g., board-room elections), each player may be both a voter and a tallier.

An election proceeds in two phases. In the first phase, the voters post their ballots, which contain the votes in encrypted form, to the bulletin board. Since the voters need not be anonymous in this scheme it is trivial to prevent double voting. Only well-formed (valid) ballots will be accepted. In the second phase, the talliers use their private keys to collectively compute the final tally corresponding with the accumulation of all valid ballots.

The protocols are as follows. Technically, each voter will act as a dealer in the PVSS scheme, where the talliers act as the participants. The initialization of the PVSS scheme is run, and we assume that each tallier A_i has registered a public key y_i .

Ballot casting A voter V casts a vote $v \in \{0, 1\}$ by running the distribution protocol for the PVSS scheme from Section 3, using a random secret value $s \in_R \mathbb{Z}_q$, and computing the value $U = G^{s+v}$. In addition, the voter constructs a proof $PROOF_U$ showing that indeed $v \in \{0, 1\}$, without revealing any information on v. $PROOF_U$ refers to the value of $C_0 = g^s$ which is published as part of the PVSS distribution protocol, and shows that:

$$\log_G U = \log_q C_0 \quad \lor \quad \log_G U = 1 + \log_q C_0$$

Such a proof can be efficiently constructed using the technique of [CDS94], see Appendix B. The ballot for voter V consists of the output of the PVSS distribution protocol, the value U, and $PROOF_U$.

Due to the public verifiability of the PVSS scheme and of $PROOF_U$, the ballots can be checked by the bulletin board when the voters submit their ballots. No involvement from the talliers is required in this stage.

Tallying Suppose voters V_j , $j=1,\ldots,m$ have all cast valid ballots. The tallying protocol uses the reconstruction protocol of the special PVSS scheme, except that we will first exploit its homomorphic property. We first accumulate all the respective encrypted shares, that is, we compute the values Y_i^* , where the index j ranges over all voters:

$$Y_i^* = \prod_{j=1}^m Y_{ij} = y_i^{\sum_{j=1}^m p_j(i)}.$$

Next, each tallier A_i applies the reconstruction protocol to the value Y_i^* , which will produce $G^{\sum_{j=1}^m p_j(0)} = G^{\sum_{j=1}^m s_j}$, due to the homomorphic property. Combining with $\prod_{j=1}^m U_j = G^{\sum_{j=1}^m s_j + v_j}$, we obtain $G^{\sum_{j=1}^m v_j}$, from which the tally $T = \sum_{j=1}^m v_j$, $0 \le T \le m$, can be computed efficiently.

Conceptually, the above election scheme fits between the schemes of [CFSY96] and [CGS97]. It achieves the same level of security with regard to universal verifiability, privacy, and robustness. Voter independence can also be achieved in the same way as in those papers. The main difference with [CGS97] is that our scheme does not require a shared-key generation protocol for a threshold decryption scheme. Such a key generation protocol needs to be executed between the talliers, and requires several rounds of interaction. An example is Pedersen's key generation protocol [Ped91], or rather the improvement by [GJKR99].

For large-scale elections, such a key generation protocol is affordable, as it reduces the work for the election itself. For elections on a smaller scale, though, the cost of shared-key generation may dominate the cost of the entire election. In particular, when the group of talliers varies from election to election, the shared-key generation phase is better avoided. Further, in small-scale elections (e.g., board-room elections), it is realistic to let each voter play the role of tallier as well. Using our new election scheme, the election consists of just the two phases described above, without relying on any interaction between the voters or between the talliers.

It is possible to instantiate the scheme of [CFSY96] to get the same type of scheme. Recall that [CFSY96] requires a private channel from each of the voters to each of the talliers. By replacing the private channels by public key encryption, the information-theoretic privacy for the voters is lost, but a scheme is obtained that also avoids the use of a shared-key generation protocol. This approach has two major shortcomings:

- The shares for the talliers are not publicly verifiable, hence we have to allow for an additional stage in the election in which the talliers may file their complaints, which must then be checked by the others, and so on, to sort out which ballots and which talliers will be part of the tallying phase. In our scheme, the talliers are not involved at all during the voting stage.
- Each tallier must decrypt its shares one by one, since there is no homomorphic property. This means that all of the encrypted shares have to be communicated to the talliers, and that the talliers have to use their private keys to decrypt each of them individually, instead of decrypting just a single accumulated share.

Given that we settle for computational privacy, we thus conclude that a PVSS-based scheme can be used for elections on a smaller scale, while [CGS97] is to be used for large-scale elections.

6 Other Applications

We present a few more examples of situations in which our PVSS scheme can be applied. We expect that PVSS offers an efficient alternative in many protocols which use VSS as a subroutine.

6.1 Threshold Binding ElGamal

In [VT97] Verheul and van Tilborg present a scheme called Binding ElGamal that may be used to enhance the strength of a public-key infrastructure depending on software key escrow. The basic scheme provides a mechanism to copy a session-key not only to the intended receiver but also to n trustees. In other words, the trustees share the session-key according to a (1, n)-threshold structure. Extensions to (t, n)-threshold scenarios are also described in [VT97]. However, in these extensions it is required that the participants first engage in a shared-key generation protocol (as in the previous section). In principle, key generation needs to be done anew each time trustees are added or removed.

Using our new PVSS scheme, we obtain a dynamic version of threshold Binding Elgamal. The sender of the message acts as the dealer in the PVSS scheme. Let y denote the public key of the intended receiver. Next to the output of the distribution protocol, the sender also publishes the value $Y = y^s$ plus a proof that $\log_g C_0 = \log_y Y$, where $C_0 = g^s$ is part of the output of the distribution protocol in Section 3. This guarantees that the intended receiver and the participants will reconstruct the same value G^s . For use as a session-key in a symmetric encryption scheme, this value may be compressed to $\mathcal{H}(G^s)$. Clearly, this variant of Binding Elgamal is open to the same kind of criticism as the basic scheme (see, e.g., [PW98]).

The advantage of this approach is that, depending on the type of message, the sender may select a group of participants that will be able to reconstruct the session-key; these participants need not engage in a prior key generation protocol. Note that we are in fact combining a (1,1)-threshold structure with a (t,n)-threshold structure (see also Appendix A).

6.2 Threshold Revocable Electronic Cash

Recent techniques to make privacy-protected cash revocable, are described in [CMS96] and [FTY96]. As a concrete example, in [CMS96] the customer must provide the bank with a value $d = y_T^{\alpha}$, where $\alpha \in_R \mathbb{Z}_q$, as part of the withdrawal protocol. Here y_T denotes the public key of the trustee. We can use our PVSS scheme to share the value d among a dynamically chosen set of trustees. Note that, as above, one can use threshold decryption with y_T representing the public key of the group as an alternative.

6.3 Threshold Software Key Escrow

Recently, Young and Yung [YY98] presented a scenario and solution for software key escrow. In their approach each user generates its own key pair, and registers it with a certification authority. The user must also encrypt shares of the secret key for a specified group of trustees. The key pair is only accepted, if the user also provides a (non-interactive) proof that the encrypted shares indeed correspond to the registered public key.

Following the model of [YY98] a possible alternative to their key escrow solution for threshold scenarios is as follows. The user generates a random private key $S = G^s$, and shares this value among the trustees using our PVSS scheme. Furthermore, the user publishes its public key $H = f^{(G^s)}$, where f is an appropriate generator (as in [Sta96]). Finally, to show that the trustees indeed receive shares of the private key $S = G^s$, the user proves knowledge of a witness s satisfying

$$H = f^{(G^s)} \wedge C_0 = g^s,$$

where C_0 is the value published in the distribution protocol in Section 3. The protocol used by Stadler in his PVSS scheme exactly matches this problem (see [Sta96, Section 3.3]). So, although we have to resort to the use of double discrete logarithms for this application, we only have to do this once (and not once per trustee!).

Decryption of an ElGamal ciphertext $(x,y) = (f^{\alpha}, H^{\alpha}m)$ is accomplished by raising x to the power G^s . We assume w.l.o.g. that the first t trustees take part in the decryption of the ciphertext. Using the fact that $G^s = \prod G^{p(i)\lambda_i}$, decryption works by setting $z_0 = x$, and letting the i-th trustee transform z_i to $z_{i+1} = z_i^{(G^{p(i)\lambda_i})} = z_i^{(S_i^{\lambda_i})}$. If desired, each trustee also produces a proof that its decryption step is correct. Again, Stadler's proof can be used for this purpose, this time to show that:

$$z_{i+1} = z_i^{(Y_i^{\lambda_i/x_i})} \land g^{\lambda_i} = y_i^{\lambda_i/x_i},$$

where y_i is the *i*-th trustee's public key and Y_i is its encrypted share of G^s .

7 Conclusion

We presented a new scheme for PVSS including several interesting applications. We stressed the virtues of the new approach, among which are its improved performance and its simplicity. Our construction hinges on the observation that it is advantageous to distribute and reconstruct a secret of the form G^s for fixed G and known random s instead of trying to reconstruct s itself. We hope to find applications of this idea in other settings too.

We have shown that many aspects play a role when selecting a secret sharing scheme for a particular application. It turns out that often we find a trade-off between the use of PVSS and the use of a threshold decryption scheme. As an example, we have considered the choice between a new PVSS based election scheme and an election scheme based on threshold decryption ([CGS97]) for large-scale elections. For elections on a smaller scale the costly key generation protocol, which is part of a threshold decryption scheme, can be avoided by using a more dynamic PVSS based approach.

Acknowledgments

Ronald Cramer, Markus Stadler, Martijn Stam, Moti Yung and the anonymous referees are gratefully acknowledged for their constructive comments.

References

- Ben87a. J. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 251–260, Berlin, 1987. Springer-Verlag.
- Ben87b. J. Benaloh. Verifiable Secret-Ballot Elections. PhD thesis, Yale University, Department of Computer Science Department, New Haven, CT, September 1987.
- Bla79. G.R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference 1979*, volume 48 of *AFIPS Conference Proceedings*, pages 313–317, 1979.
- Bri89. E. F. Brickell. Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 9:105–113, 1989.
- BY86. J. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. In *Proc. 5th ACM Symposium on Principles of Distributed Computing (PODC '86)*, pages 52–62, New York, 1986. A.C.M.
- CDM99. R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret sharing scheme, 1999. Manuscript.
- CDS94. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.
- CF85. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pages 372–382. IEEE Computer Society, 1985.
- CFSY96. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret ballot elections with linear work. In *Advances in Cryptology—EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83, Berlin, 1996. Springer-Verlag.
- CGMA85. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pages 383–395. IEEE Computer Society, 1985.
- CGS97. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology—EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118, Berlin, 1997. Springer-Verlag.
- CMS96. J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In *Computer Security ESORICS 96*, volume 1146 of *Lecture Notes in Computer Science*, pages 33–43, Berlin, 1996. Springer-Verlag.
- CP93. D. Chaum and T. P. Pedersen. Transferred cash grows in size. In Advances in Cryptology—EUROCRYPT '92, volume 658 of Lecture Notes in Computer Science, pages 390–407, Berlin, 1993. Springer-Verlag.
- Fel87. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. 28th IEEE Symposium on Foundations of Computer Science (FOCS '87)*, pages 427–437. IEEE Computer Society, 1987.
- FO98. E. Fujisaki and T. Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In *Advances in*

- Cryptology—EUROCRYPT '98, volume 1403 of Lecture Notes in Computer Science, pages 32–46, Berlin, 1998. Springer-Verlag.
- FTY96. Y. Frankel, Y. Tsiounis, and M. Yung. "Indirect discourse proofs": Achieving efficient fair off-line e-cash. In *Advances in Cryptology—ASIACRYPT* '96, volume 1163 of *Lecture Notes in Computer Science*, pages 286–300, Berlin, 1996. Springer-Verlag.
- GJKR99. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310, Berlin, 1999. Springer-Verlag.
- KW93. M. Karchmer and A. Wigderson. On span programs. In Proceedings of the Eighth Annual Structure in Complexity Theory Conference, pages 102–111.
 IEEE Computer Society Press, 1993.
- Ped91. T. Pedersen. A threshold cryptosystem without a trusted party. In Advances in Cryptology—EUROCRYPT '91, volume 547 of Lecture Notes in Computer Science, pages 522–526, Berlin, 1991. Springer-Verlag.
- Ped92a. T. P. Pedersen. Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem. PhD thesis, Aarhus University, Computer Science Department, Aarhus, Denmark, March 1992.
- Ped
92b. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Advances in Cryptology—CRYPTO '91, volume 576 of Lecture Notes in Computer Science, pages 129–140, Berlin, 1992. Springer-Verlag.
- PW98. B. Pfitzmann and M. Waidner. How to break fraud-detectable key recovery. Operating Systems Review, 32(1):23–28, 1998.
- Sha79. A. Shamir. How to share a secret. Communications of the ACM, 22(11):612–613, 1979.
- Sta96. M. Stadler. Publicly verifiable secret sharing. In Advances in Cryptology— EUROCRYPT '96, volume 1070 of Lecture Notes in Computer Science, pages 190–199, Berlin, 1996. Springer-Verlag.
- VT97. E. Verheul and H. van Tilborg. Binding ElGamal: A fraud-detectable alternative to key-escrow proposals. In Advances in Cryptology—EUROCRYPT '97, volume 1233 of Lecture Notes in Computer Science, pages 119–133, Berlin, 1997. Springer-Verlag.
- YY98. A. Young and M. Yung. Auto-recoverable auto-certifiable cryptosystems. In Advances in Cryptology—EUROCRYPT '98, volume 1403 of Lecture Notes in Computer Science, pages 17–31, Berlin, 1998. Springer-Verlag.

A Extension to any Linear Secret Sharing Scheme

In Section 3 we have described a PVSS scheme based on Shamir's secret sharing scheme. However, the construction readily extends to any linear secret sharing scheme. A secret sharing scheme is linear if the dealer and the participants only use linear operations to compute with the shares and the secret. Below, we present a PVSS scheme based on Brickell's vector space construction [Bri89]. A more general class of linear secret sharing schemes is formed by the class of monotone span programs [KW93], which turns out to play an important role in the general context of secure multi-party computation (see [CDM99]).

Let Γ be a monotone access structure on $\{1, \ldots, n\}$ and consider the following secret sharing scheme with secrets and shares in \mathbb{Z}_q , where q is prime. Suppose we have constructed an $n \times d$ matrix M, for some d, satisfying for all $B \subseteq \{1, \ldots, n\}$:

$$B \in \Gamma \quad \equiv \quad e_1 \in \{c \ M_B \mid c \in \mathbf{Z}_q^{|B|}\},\tag{1}$$

where $e_1 = (1, 0, ..., 0)$, and M_B denotes the submatrix consisting of all rows indexed by B. Hence, a set B of participants is qualified just when e_1 is contained in the span of the rows of M_B .

To distribute a random secret s, the dealer picks a random column vector $a \in \mathbb{Z}_q^d$, sets $s = a_1$, and computes the share for participant P_i as $s_i = M_i a$. As before, the dealer publishes the commitments $C_j = g^{a_j}$, for $0 \le j < t$, and the encrypted shares $Y_i = y_i^{s_i}$, for $1 \le i \le n$. To prove the consistency of the encrypted shares, the dealer proves that $\log_g X_i = \log_{y_i} Y_i$, where $X_i = g^{s_i}$ can be computed by anyone (due to the linearity):

$$X_i = \prod_{j=1}^{d} (C_j)^{M_{ij}} = g^{\sum_{j=1}^{d} M_{ij} a_j} = g^{s_i}$$

As before, for reconstruction the participants decrypt their shares to obtain $\{G^{s_i}\}_{i\in B}$, for some $B\in \Gamma$. Using a vector c for which $e_1=c$ M_B , which exists on account of (1), we then have (again due to the linearity):

$$\prod_{i \in B} (G^{s_i})^{c_i} = G^{\sum_{i \in B} (M_i a) c_i} = G^{(\sum_{i \in B} c_i M_i) a} = G^{e_1 a} = G^s.$$

Clearly, the resulting schemes are both simple and efficient. In general, only a single commitment C_j is required per random value chosen by the dealer, and one encryption Y_i per share s_i plus a response r_i for the consistency proof. Note that the construction also works for secret sharing schemes in which participants may receive multiple shares.

B Description of $PROOF_U$

See Section 5. In order to prove that U is well-formed the prover must convince the verifier that there exists an s such that $C_0 = g^s$ and $U = G^{s+v}$ with $v \in \{0,1\}$. The protocol runs as follows:

- 1. The prover sets $a_v = g^w$ and $b_v = G^w$ for random $w \in_R \mathbb{Z}_q$. The prover also sets $a_{1-v} = g^{r_{1-v}}C_0^{d_{1-v}}$ and $b_{1-v} = G^{r_{1-v}}(U/G^{1-v})^{d_{1-v}}$, for random $r_{1-v}, d_{1-v} \in_R \mathbb{Z}_q$. The prover sends a_0, b_0, a_1, b_1 in this order to the verifier.
- 2. The verifier sends a random challenge $c \in_R \mathbb{Z}_q$ to the prover.
- 3. The prover sets $d_v = c d_{1-v} \pmod{q}$ and $r_v = w sd_v \pmod{q}$, and sends d_0, r_0, d_1, r_1 in this order to the verifier.
- 4. The verifier checks that $c = d_0 + d_1 \pmod{q}$ and that $a_0 = g^{r_0} C_0^{d_0}$, $b_0 = G^{r_0} U^{d_0}$, $a_1 = g^{r_1} C_0^{d_1}$, and $b_1 = G^{r_1} (U/G)^{d_1}$.

Clearly, the proof is honest verifier zero-knowledge, hence its non-interactive version releases no information on v in the random oracle model.

Truncated Differentials and Skipjack

Lars R. Knudsen¹, M.J.B. Robshaw², and David Wagner³

- Department of Informatics, University of Bergen, N-5020 Bergen, Norway larsr@ii.uib.no
 - ² RSA Laboratories, 2955 Campus Drive, San Mateo, CA 94403, USA matt@rsa.com
 - ³ University of California Berkeley, Soda Hall, Berkeley, CA 94720, USA daw@cs.berkeley.edu

Abstract. We consider a range of attacks on reduced-round variants of the block cipher Skipjack. In particular we concentrate on the role of truncated differentials and consider what insight they give us into the design and long-term security of Skipjack. An attack on the full 32 rounds of Skipjack remains elusive. However we give attacks on the first 16 rounds of Skipjack that can efficiently recover the key with about 2¹⁷ chosen plaintexts and an attack on the middle sixteen rounds of Skipjack which recovers the secret key using only two chosen plaintexts. Several high-probability truncated differentials are presented the existence of which might best be described as surprising. Most notably, we show that the techniques used by Biham et al. can be presented in terms of truncated differentials and that there exists a 24-round truncated differential that holds with probability one.

1 Introduction

Skipjack is a 64-bit block cipher that is used in the Clipper Chip [11,12] and was recently made public by the NSA [15,16]. The length of the user-supplied key suggests that like other cryptographic proposals from the U.S. government [13,14] the security level is intended to be 80 bits. Skipjack is a remarkably simple cipher and one interesting feature is the use of two different types of rounds. These are referred to as A-rounds and B-rounds and encryption with Skipjack consists of first applying eight A-rounds, then eight B-rounds, once again eight A-rounds and finally eight B-rounds.

The simplicity of Skipjack alone makes it an interesting cipher to study. However if we also recall the speculation and widespread distrust with which the cipher was first received [11,12] then this once-secret cipher becomes particularly intriguing. In this paper we will consider some of the structural properties of Skipjack. In particular we note that the simple rounds of Skipjack seem to be particularly amenable to analysis using truncated differentials [7]. We will provide details of some particularly effective attacks on reduced-round versions of Skipjack and we will consider the applicability of these and other potentially more powerful attacks to an analysis of the full cipher.

A preliminary version of a report into the security of Skipjack was published on July 28, 1993 [9]. Written by five eminent cryptographers, the report reveals that while Skipjack was designed using techniques that date back more than forty years, Skipjack itself was initially designed in 1987. Since the cipher and design details were classified at the time of writing, the authors of the report were clearly restrained in what they could say. However, one phrase in the report is particularly interesting. There it is claimed that "[The design process] eliminated properties that could be indicative of vulnerabilities" [9]. In this paper we demonstrate that, in our opinion, this design goal was not attained. While we find no way to exploit the structural features that we highlight in a direct attack on the full Skipjack, we feel that the presence of these features could well be indicative of vulnerabilities. With more study, they could lead others towards an exploitable weakness in the cipher.

2 Description of Skipjack and other work

The 64-bit input block of Skipjack is split into four words of 16 bits. At the time of its initial design (1987) this approach was perhaps somewhat uncommon though RC2 [8] adopts a similar structure. In each round of Skipjack one of the words passes through a keyed permutation which we denote by G, and at most two words are modified during a single round. The function G has the structure of a four-round, byte-wise Feistel network. When needed, we will denote the round function (which uses a fixed, byte-wise substitution table S) by F. A counter, which is incremented at each round of encryption, is also used though it will be ignored throughout this paper since it has no cryptographic impact on our work. The rounds are illustrated in Figure 1.

The user-supplied key features during the G transformation. At each round four bytes of the 10 bytes of key material are used, with one byte being used at each step of the mini-Feistel network contained within G. If we denote the key by $k_0
ldots k_0
ldots$

A first analysis by Biham et al. [1] studied some of the detailed properties of G and in particular some of the properties of the substitution table S. This provided a first description of some differential [6] and linear [10] cryptanalytic attacks on reduced-round versions of Skipjack. It was shown that reducing Skipjack to consist of the first 16 rounds (eight A-rounds followed by eight B-rounds) allowed one to mount a differential attack requiring about 2^{55} chosen plaintexts 1 .

Independently of the authors of this paper, Biham et al. [2,3] also considered the role of truncated differentials in Skipjack and some variants. All that is important for such attacks to be mounted is that the function G be a permutation. Further details about G (and therefore of the substitution box S) are

¹ It is important to note that this attack required that the key schedule be treated in a way that seems to conflicts with its intended use in the full cipher.

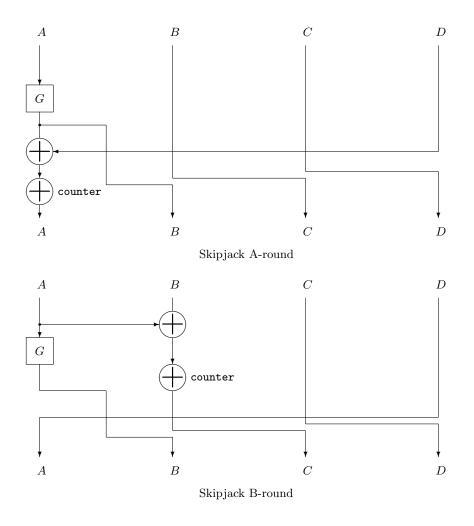


Fig. 1. The two rounds used in Skipjack. The counter is encrypted at each round and while it is included for completeness, it has no cryptanalytic significance with regards to the attacks in this paper.

immaterial. Most recently Biham et al. [5] derived attacks that are faster than exhaustive search for the key if Skipjack is reduced by at least one round. In this paper we consider alternative enhancements which offer interesting insights into the design of Skipjack. Currently these seem to be less effective than other attacks but we observe that there are opportunities for improvement and we outline promising avenues for further work that remain unexplored.

3 Truncated differentials of Skipjack

In a typical differential attack, the attacker chooses two plaintexts with a particular difference between them. During the encryption process the aim is to predict with some probability how the difference between these two quantities evolves. When the attacker is able to predict the difference towards the end of the encryption process with a sufficiently high probability, information about the user-supplied key can sometimes be derived.

When using truncated differentials, instead of trying to predict the evolution of some difference across an entire block, the cryptanalyst attempts to predict the difference across some fraction of this block. With Skipjack it is very natural to consider the difference across the four 16-bit words as we will now demonstrate. Let a, b, \ldots, h denote any non-zero value to the difference² in a 16-bit word. We use r_A to denote an A-round and r_B to denote a B-round. One useful truncated differential characteristic allows us to cover the first 16 rounds of Skipjack:

$$(a, b, 0, c) \xrightarrow{8r_A} (e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0),$$
 (1)

The probability of the differential is 2^{-32} since $(a, b, 0, c) \xrightarrow{4r_A} (0, d, 0, 0)$ with probability 2^{-32} , $(0, d, 0, 0) \xrightarrow{4r_A} (e, e, 0, 0)$ always holds, and the last eight rounds of the characteristic $(e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0)$ always holds. This differential will be useful to us in Section 4.1 where it is shown how to break the first 16 rounds of Skipjack with 2^{17} chosen plaintexts.

There are other interesting truncated differentials for Skipjack. The truncated differential (1) contains a truncated differential over eight B-rounds which holds with probability one. We found that there are at least two other truncated differentials over eight B-rounds which hold with the same probability. They are

$$(0,0,a,0) \xrightarrow{8r_B} (b,0,c,d)$$
 and $(0,a,0,0) \xrightarrow{8r_B} (0,b,c,d)$.

It is possible to add another four A-rounds to the latter differential while retaining the fact that the truncated differential holds with probability one. Thus, one gets the following twelve-round truncated differential with probability one

$$(0, a, 0, 0) \xrightarrow{8r_B} (0, b, c, d) \xrightarrow{4r_A} (h, h, f, g).$$
 (2)

In Section 4.2 we will use this truncated differential to mount a particularly efficient truncated differential attack on the middle 16 rounds of Skipjack.

While a 12-round truncated differential with probability one seems remarkable enough, there is more and these results are described in Section 3.1. We also highlight some practical difficulties when using truncated differentials in Section 3.2 and we describe the semi-exhaustive search that we used to find these differentials in Section 3.3. We note here that the 16-round truncated differential (1) given above is indeed the best truncated differential for the first 16 rounds of Skipjack.

² While the most useful notion of difference can change depending on the cipher in question, for Skipjack we use bitwise exclusive-or.

3.1 Long truncated differentials

At least one truncated differential gives nontrivial information about the ciphertexts after 17 rounds of encryption. It goes through four A-rounds, eight B-rounds and five A-rounds and has the following form:

$$(0, a, 0, 0) \xrightarrow{4r_A} (b, b, 0, 0) \xrightarrow{8r_B} (c, d, e, 0) \xrightarrow{5r_A} (f, g, h, i),$$

where $f \neq g$ and h and i can take any values. A variant of Skipjack reduced to these 17 rounds can be distinguished from a randomly chosen permutation using only about $\sqrt{2} \cdot 2^8$ chosen plaintexts³.

Even more remarkably there are truncated differentials which give non-trivial information about the ciphertexts after up to 24 rounds of encryption. It is interesting to compare the following 24-round truncated differential with the 24-round "impossible differential" of Biham et al. [5]. They are identical, though the differential described here will be explained in the classical top-down fashion.

First consider the following 12-round differential that also features in [5]. The words a, b, c, d, e can be arbitrary nonzero values.

$$(0, a, 0, 0) \xrightarrow{4r_A} (b, b, 0, 0) \xrightarrow{8r_B} (c, d, e, 0)$$
(3)

The differential can be concatenated with the following differential over 8 Arounds and 4 B-rounds.

$$(c, d, e, 0) \xrightarrow{8r_A} (j, k, l, m) \xrightarrow{4r_B} (r, s, t, u)$$
 (4)

If we are careful to track how the differential evolves, we are able to place conditions on different words of the differential even if they are identified as being non-zero. A pair of inputs have equal values in the fourth word, but different values in the other three. The conditions at each round of the evolution of truncated differential (4) are given in Table 1. Note, as an example, that after the second A-round $f \neq e$ since $f = c \oplus e$ and $c \neq 0$. Likewise, after the fourth A-round $(i,g) \neq (0,0)$. To see this, note that in the preceding round $h \neq g$, since $b \neq 0$. But $h \neq g$ implies that $(h,g) \neq (0,0)$ and $(i,g) \neq (0,0)$ since $i = 0 \Leftrightarrow h = 0$.

We can show (see Table 1) that the three rightmost words at the end of the last 12 rounds of the 24-round truncated differential cannot all be zero. Suppose to the contrary that $w=0, \ \gamma=0, \ \text{and} \ \beta=0$. This implies $k=0 \Rightarrow v=0 \Rightarrow p=0 \Rightarrow u=0 \Rightarrow m=0, \ \text{and} \ \text{we have a contradiction since} \ (m,k)\neq (0,0).$ Altogether, this yields a 24-round truncated differential, where the differences in the three rightmost words of the ciphertexts cannot all be zero.

³ To see this choose a pool of different plaintexts with equal values in the first, third and fourth words. Compute the exclusive-or of the first two words of all ciphertexts and look for a match in these values. Such a match will not be found for the Skipjack variant, but for a randomly chosen permutation a match is found with probability 2^{-16} .

Round	Difference	Properties
	(a, b, c, 0)	a, b, c nonzero
A1:	(d,d,b,c)	b, c, d nonzero
A2:	(f,e,d,b)	e, d, b nonzero, $f \neq e$
A3:	(h,g,e,d)	e, d nonzero, $h \neq g$
A4:	(j,i,g,e)	$e \neq 0, (i, g) \neq (0, 0), j \neq i$
A5:	(l,k,i,g)	$(k,i) \neq (0,0), (i,g) \neq (0,0), l \neq k$
A6:	(n,m,k,i)	$(m,k) \neq (0,0)$
A7:	(q, p, m, k)	$(m,k) \neq (0,0)$
A8:	(s,r,p,m)	$s = k \oplus r$
B1:	(m,t,k,p)	$(m,k) \neq (0,0)$
B2:	(p, u, α, k)	$\alpha = m \oplus t$
B3:	(k,v,eta,lpha)	$\beta = p \oplus u$
B4:	$(\alpha, w, \gamma, \beta)$	$\gamma = k \oplus v$

Table 1. The last 12 rounds of the 24-round truncated differential.

3.2 Important practical details

Before proceeding it is worth highlighting two important features of a truncated differential if we wish to use it directly in an attack.

FILTERING. After accumulating what might be a vast amount of chosen plaintext-ciphertext pairs in an attack, the cryptanalyst needs to throw away as much erroneous data (pairs that do not follow the differential as intended) as possible. This is done by filtering. With the truncated differentials we consider, the structure we use for filtering is the presence of a zero difference in some word. In the 16-round attack of Section 4.1, the expected difference in the ciphertexts is (g, h, f, 0), which means that only pairs of ciphertexts with equal fourth words will be left after filtering. The more zeros in the expected output difference, the greater the number of wrong pairs that can be filtered before starting to extract key material.

Counting. The second feature that is important to consider is where, in some input and output difference, the non-zero differences lie. While some truncated differentials might initially appear to be useful to the cryptanalyst, it is not always possible to extract key information. One example is the following truncated differential $(0, a, b, 0) \xrightarrow{8r_A} (c, d, e, 0) \xrightarrow{8r_B} (0, f, g, h)$ which holds with probability 2^{-32} . When using this differential in an attack it passes over the first round of encryption with probability one and it is not possible to distinguish the correct first-round subkey from the wrong ones.

The semi-exhaustive search described in Section 3.3 was completed for truncated differentials of the full 32-round Skipjack. The search revealed several truncated differentials of probability 2^{-64} . However for all of these it seems impossible to search for keys in both the first and the last round as would be needed to directly mount an attack.

3.3 The search for truncated differentials

The semi-exhaustive search for truncated differentials was done in the following manner. Represent each 16-bit word in Skipjack by a single bit and so four bits will be used to represent the internal state of the four 16-bit words in Skipjack. A zero in the $i^{\rm th}$ bit indicates that there is no difference in the values of the $i^{\rm th}$ of the pair of data that follow the differential. The value one is used to indicate a non-zero difference which results from the two words having different values. We can then specify a set of rules that describes the "encryption" of the four words of difference through the A-rounds and through the B-rounds. It is easy in this way to do a complete search for any number of A-rounds and similarly for any number of B-rounds.

When combining A-rounds with B-rounds as required in Skipjack an extra "rule" is required. In the case where an A-round is followed by a B-round and where the output difference of the A-round has nonzero values assigned to the first two words, one needs to know if the difference in the first word is equal to the difference in the second word. This is of vital importance in the calculation of the probability of the differential in the B-round. However it is also easy to incorporate this consideration as a part of the search, since the differences in the two first output words from an A-round will be equal if, and only if, the fourth words of the inputs to the A-round are equal. Since no extra "rules" are needed in the transition from a B-round to an A-round, one can find truncated differentials for any number of A- and B-rounds.

In the following we report several findings of the search algorithm. In variants starting with eight B-rounds followed by eight A-rounds the following truncated differential

$$(0, a, 0, 0) \xrightarrow{4r_B} (0, b, c, 0) \xrightarrow{4r_B} (0, d, e, f) \xrightarrow{4r_A} (0, 0, q, h) \xrightarrow{4r_A} (0, 0, 0, i)$$

has component-wise probabilities of 1, 1, 2^{-16} , and 2^{-16} for the component four-round differentials respectively. Totally, the differential has probability 2^{-32} and a pair of texts following the differential can be found by taking all pairs generated from a pool of about 2^{17} chosen plaintext values. This makes it possible to effectively distinguish this variant of 16-round Skipjack from a random permutation using only around 2^{17} chosen plaintexts. For a random permutation two ciphertexts with equal values in the first three words (as in the above differential) occur with probability 2^{-48} and such a pair would normally be expected to occur after generating around $\sqrt{2} \cdot 2^{24}$ values.

The search revealed several truncated differentials for the full Skipjack with probability 2^{-64} . One example is the following differential where the words a, \ldots, m can take any nonzero values.

$$(0, a, b, c) \xrightarrow{8r_A} (0, 0, 0, d) \xrightarrow{8r_B} (0, e, f, g) \xrightarrow{8r_A} (h, h, i, j) \xrightarrow{8r_B} (0, k, l, m),$$

This differential allows only for a very limited amount of filtering since only the form of the leftmost word of the ciphertext is restricted. (For all the 32-round truncated differentials with probability 2^{-64} that we have identified, only one of

the words in the expected output differences is zero.) Furthermore, the leftmost word of the plaintext difference in all cases is zero, which means that key material cannot be extracted from analysis of the first round since all possible subkeys are equally likely. Thus, these differentials do not seem to be useful in an attack. However it is possible, at least theoretically, to mount an attack on the last 28 of the 32 rounds of Skipjack as we will show later.

4 Attacks using truncated differentials

4.1 The first sixteen rounds

We start with a truncated differential attack on the first 16 rounds of Skipjack that requires only 2^{17} chosen plaintexts and about 2^{34} – 2^{49} time for the analysis. The range in the computational complexity comes from whether we treat the first and last round subkeys as independent or not.

We note that truncated differential cryptanalysis allows for significant improvements over an ordinary differential attack [1] due to two effects. First, the probability of the differential is sharply increased from $2^{-52.1}$, which was the probability of the differential [1] used in the conventional differential attack, to 2^{-32} . Second, the truncated differential allows us to extract more usable plaintext pairs from fewer chosen plaintexts because there is additional freedom in the construction of what are termed structures [6].

The attack uses the truncated differential (1) for the first 16 rounds of Skipjack. To generate suitable pairs for such a differential we choose 2^{17} plaintexts where the third words are fixed and obtain the corresponding ciphertexts. From these plaintexts one can form about 2^{33} pairs with the desired starting difference. With a high probability two right pairs will follow the truncated differential. Observing that the rightmost word has zero difference, we can immediately filter out many wrong pairs before moving on to the next stage of the analysis with 2^{17} pairs of data. In this second phase we will extract key material from the first and sixteenth rounds but the analysis will differ depending on whether the subkeys used in the outer two rounds are the same or different.

INDEPENDENT SUBKEYS. Here we treat the case where the subkeys used in the first and 16th rounds are independently chosen. This seems more true to the intent of the Skipjack designers and is perhaps a better reflection of the style of attack that is needed for the full 32-round version of Skipjack.

Using the same truncated differential as before, each pair that survives filtering will suggest 2^{16} values for the four key bytes in the first round, and 2^{16} values for the four key bytes in the last round. It is possible to find these 2^{17} suggested values with offline work comparable to about 2^{17} G-box computations [4,5]. (The trick is to use a precomputed table which, given differences y, z, allows us to find input x such that $F(x) \oplus F(x \oplus y) = z$ with one table lookup. We guess k_2, k_3 , decrypt up by two layers of the G-box, and use the precomputed table to recover k_0, k_1 , noting that z is known from the G-box input difference and y is known as a result of decrypting up two layers.) In total, we find that after filtering each remaining pair suggests about 2^{32} values for the eight key bytes used in the first and $16^{\rm th}$ rounds. Naively we could simply count on those 64 key bits and look for a counter whose value exceeds one. By the birthday paradox, only about $2^{2\times 49}/2^{64+1}=2^{33}$ wrong key values would remain, and each suggested value for the eight key bytes could be tested by exhaustive search over the remaining two unknown bytes. Thus, we could recover the key with about $(2^{17}\times 2^{32})+(2^{33}\times 2^{16})=2^{50}$ work but the need for 2^{64} counters makes this approach totally impractical⁴. Instead we suggest the following technique.

Examine the plaintext pairs two at a time. For each two plaintext pairs use the calculation of G in the $16^{\rm th}$ round to recover a list of possible values for the subkey. On average we expect to find about one possible subkey value. Similarly, the G computation in the first round is used to recover a possible value for another four key bytes. The suggested value for these eight key bytes can then be tested by exhaustive search over the remaining two unknown key bytes. There are about $2^{17} \cdot (2^{17}-1)/2 \approx 2^{33}$ ways to choose two plaintext pairs, and each one requires about 2^{16} work, so with work equivalent to about 2^{49} encryptions we can recover the key.

The computational complexity could be reduced using alternative techniques if more texts are available. We can form 2^{37} plaintext pairs from 2^{19} chosen plaintexts, and count on the last-round subkey. About 2^{21} pairs survive filtering and so incrementing the counters requires work equivalent to about 2^{37} computations of G. The right counter will be suggested about $2^5 + 2^5$ times, whereas the wrong counter will be suggested 2^5 times on average (with standard deviation $2^{2.5}$). Only about 32 wrong counters will exceed their mean value by $2^{2.5} \approx 5.66$ standard deviations or more, so only about 33 values for the last-round subkey will survive. Similarly, we can find 33 possibilities for the first-round subkey with another 2^{37} computations of G, so after time equivalent to $(2^{37} + 2^{37})/16 = 2^{34}$ trial encryptions we can recover 33^2 possibilities for 64 key bits. Finally, those 33^2 possibilities can be tested with an exhaustive search over the remaining two unknown key bytes. The total computational complexity is equivalent to $2^{34} + 33^2 \times 2^{16} \approx 2^{34}$ trial encryptions with 2^{19} chosen plaintexts and 2^{32} space.

DEPENDENT SUBKEYS. When the subkeys used in the first and $16^{\rm th}$ rounds are the same⁵, several optimizations may be applied to the truncated differential attack. In this case, the total amount of offline work required for the attack is roughly comparable to that needed for 2^{34} offline trial encryptions.

4.2 The middle sixteen rounds

It is interesting to observe that there is a very efficient way to break the middle 16 rounds of Skipjack, i.e. a version of Skipjack consisting of eight B-rounds

Space requirements can be reduced to about $2^{49} \times 8 = 2^{52}$ bytes by using a hash table or sorted list to store the suggested key values, but this is still too large.

⁵ This holds for Skipjack. However, we feel that this is somewhat artificial since it is highly likely that any designers of such a Skipjack variant would change the key schedule to avoid this eventuality.

followed by eight A-rounds. Of course Skipjack has many more rounds than the sixteen we are attacking here, but our work is interesting for two reasons.

First it demonstrates that there is an asymmetry in how the A-rounds and the B-rounds might combine together to resist the attacker. This might help provide some insight into the design rationale behind Skipjack. Second, the attack outlined makes use of the structure of the G computation, and most importantly, of the internal Feistel structure. As in the earlier attacks, the S-box itself is completely immaterial to our discussions, but the byte-wise nature of the G computation provides a real benefit to the attacker. It is possible that attacks depending on the word-oriented structure of Skipjack could be aided by also considering the byte-oriented structure of the G computation. As a demonstration of this, we show that with two chosen texts, we can break this reduced cipher with work equivalent to about 2^{47} trial encryptions; with three chosen texts, the complexity of the attack drops to about 2^{30} encryptions. This is surprisingly close to Skipjack's unicity distance (1.25 texts).

We shall number the rounds from 1 to 16, so that the first round uses k_0, \ldots, k_3 and so on. In this attack, we use the 12-round truncated differential (2) of probability one to cover rounds 1 to 12 of the reduced cipher.

First we obtain n independent pairs following the truncated differential by making n+1 chosen-plaintext queries⁶ with the first, third, and fourth words of the input fixed. The rest of this section describes how to analyze those n pairs. We will describe our attack in general terms, leaving the number n of pairs unspecified until the end. Afterwards, we will optimize over n to obtain the best possible results.

The analysis consists of seven phases. In each phase, we recover some portion of the key material, either by guessing or by deriving it from known quantities inside the cipher. We describe each of the seven phases in turn.

- 1. Guess k_0, \ldots, k_3 . For each pair, peel off the 16th round to learn the value of h that this key guess suggests.
- 2. Recover k_9 . A naive approach is to simply guess k_9 ; reversing three layers of the computation of G in round 13 (using k_1, k_0, k_9) will give the right half (low byte) of h in each pair if our guess for $k_{9...3}$ was correct. This gives a filtering condition on 8n bits. In practice, this can be implemented efficiently using a precomputed lookup table; see Section 4.1 or [4,5] for more details. With proper implementation, the work factor of this phase will be about 2^{32} , and we expect 2^{40-8n} values of $k_{9...3}$ to remain.
- 3. Recover k_8 . We can use the same technique as in the second phase, this time reversing a fourth layer of the G transformation in round 13. We predict that the exclusive-or of the values obtained should be the same as the left half (high byte) of h in each pair if our guess was correct. This gives a filtering

⁶ One could use structures to obtain n pairs from $\sqrt{2n} + 1$ queries, but the resulting pairs would not be independent, and we do not expect the extra "dependent" pairs to provide any useful extra information. Furthermore, typically we only need n = 2 pairs, so the difference would be negligible in any case.

condition on 8n bits, so 2^{48-16n} possibilities for k_8, \ldots, k_3 will remain. With proper implementation, this phase takes about 2^{40-8n} work.

- 4. Guess k_4 and k_5 . Now decrypt through the computation of G in round 14 (using k_2, \ldots, k_5) to learn g. This will suggest 2^{64-16n} values for k_8, \ldots, k_5 , with a similar work factor.
- 5. Recover k_7 . The outputs of the G transformation in round 10 are now known, and the inputs have known difference a. We can decrypt two layers of G in round 10 (using k_8 and k_9), and then derive k_7 (which is used in the next layer) with a precomputed lookup table, as above. With proper implementation, this phase takes 2^{64-16n} work, and we expect that about 2^{72-24n} possibilities for k_7, \ldots, k_5 will remain at the conclusion of this phase.
- 6. Recover k_6 . Complete the analysis of the computation of G in round 10 by deriving k_6 from its known outputs and its known input difference a. With proper implementation, this phase takes 2^{72-24n} simple operations, and about 2^{80-32n} suggested values for the entire key k_0, \ldots, k_9 will be left.
- 7. Check suggested values. We can check each suggested value for the key in any of a number of ways. One simple way is to do a full trial decryption on a few of the texts. Alternately, one could encrypt through the G transformations in rounds two, three, and six to check the result against the known input to G in round 10. This will require only four G computations and thus can be quite a bit faster than a full trial decryption. We expect that this final phase will quickly eliminate all incorrect keys.

The work required is about

$$2^{32} + 2^{32} + 2^{40-8n} + 2^{64-16n} + 2^{64-16n} + 2^{72-24n} + 4 \times 2^{80-32n}$$

simple operations. For n=1 this gives 2^{51} steps and 2^{34} steps for n=2. Of course, each step requires just a single G computation (often quite a bit less), so this is equivalent to about 2^{47} (respectively 2^{30}) trial encryptions. The result is a very sharp attack against the middle 16 rounds of Skipjack.

4.3 The last twenty-eight rounds

In this section we consider Skipjack reduced to the last 28 rounds and the following 28-round differential:

$$(a,b,0,c) \xrightarrow{4r_A} (d,e,0,0) \xrightarrow{8r_B} (f,g,0,h) \xrightarrow{8r_A} (i,i,0,0) \xrightarrow{8r_B} (j,k,l,0),$$

where $(a, b, 0, c) \longrightarrow (d, e, 0, 0)$ is a four-round differential that starts in the fifth round, ends in the eighth round, and holds with probability 2^{-16} . The following eight-round differential has probability 2^{-16} , the next has probability 2^{-32} , and the final eight-round differential has probability 1. This gives a truncated differential over the last 28 rounds of Skipjack which holds with probability 2^{-64} .

To start the attack, choose 2^{41} plaintexts where the values of the third words are fixed to some arbitrary value. From these plaintexts we can form about 2^{81} pairs of which 2^{17} will be expected to follow the specified differential. Using the

rightmost word of the ciphertexts we can filter out wrong pairs, leaving 2^{65} pairs. The extraction of key material that follows is similar to that given in Section 4.1.

INDEPENDENT SUBKEYS. First we assume that the keys in the first round of the differential (the fifth round of Skipjack) are independent of the keys in the last round. For each surviving pair we check which keys in the last round result in a difference that follows the differential after decryption by one round. About 2¹⁶ values of the 32-bit key will be suggested by this test for each pair. Similarly, for each surviving pair we check which keys result in differences that follow the differential after encryption by one round. With an efficient implementation, the suggested key values can be found in time comparable to 2^{17} evaluations of G (see Section 4.1 or [4,5]). Overall we will find that $2^{65+16+16} = 2^{97}$ values for 64 bits of key material will be suggested. The expected value of the counter for a wrong value of the key is 2^{33} , whereas the expected value of the counter for the correct value of the key will be $2^{33} + 2^{17}$ since each of the 2^{17} right pairs will include the correct key value among the set of 2^{32} values suggested. This would mean that with a high probability the correct value of the key is among the 16% most suggested values. The total time for the analysis stage of this attack amounts to $2^{65+17} = 2^{82}$ G computations, a work effort that is equivalent to about 2⁷⁷ encryptions. Thus, this attack is just faster than an exhaustive search for the key but the work effort required and the need for 2^{64} counters makes the attack totally impractical.

DEPENDENT SUBKEYS. If we assume that this reduced-round variant of Skipjack uses the key schedule specified in Skipjack then the attack will improve. The subkeys used in the fourth round are key bytes k_2 , k_3 , k_4 , and k_5 . The subkeys used in the last round are key bytes k_4 , k_5 , k_6 , and k_7 . The two sets of subkeys have a total entropy of only 48 bits. When taking this into account analysis of the data will suggest $2^{65+16}=2^{81}$ values for a 48-bit key. The rest of the analysis is the same but the memory requirements have been reduced to 2^{48} counters.

We anticipate that similar attacks on Skipjack with fewer than 28 rounds will be much more efficient and that they can be used to find more information about the secret key. Furthermore, it might be possible to attack versions of Skipjack by counting on 64 key bits when the subkeys in the first two rounds and the last two rounds together have an entropy of 64 bits. We note that such a fortuitous key-scheduling coincidence occurs in the full 32-round Skipjack cipher.

5 Boomerang attacks

Here we consider the feasibility of boomerang attacks [17] on reduced-round variants of Skipjack. Boomerang attacks may be considered to be a close relative of miss-in-the-middle attacks [5], although these techniques were developed independently. Boomerang attacks on Skipjack are interesting because they allow us to improve on some of the existing miss-in-the-middle attacks by a factor of $2^{3.5}$ – $2^{8.5}$. However, miss-in-the-middle attacks currently penetrate more rounds of Skipjack than boomerang attacks.

Boomerang attacks are chosen-plaintext, adaptive chosen-ciphertext attacks that work from the outside in. They use differential techniques to create a quartet structure inside the cipher by working from both ends of a cipher (the plaintext and ciphertext inputs) towards the middle. This quartet consists of four plaintexts P, P', Q, Q', along with their respective ciphertexts C, C', D, D' chosen as follows. We use a truncated differential $\Delta \to \Delta^*$ for the first half of the cipher, as well as the truncated differential $\nabla \to \nabla^*$ for the inverse of the last half of the cipher. The cryptanalyst picks P, P' so that $P \oplus P' \in \Delta$, encrypts to obtain C, C', then picks D, D' so that $C \oplus D \in \nabla$ and $C' \oplus D' \in \nabla$. The cryptanalyst then asks for the decryption of D, D' to obtain Q, Q'. We hope that the pair P, P' follows the differential $\Delta \to \Delta^*$, and that the pair C, D and the pair C', D'both follow the differential $\nabla \to \nabla^*$. If so, we have a quartet structure halfway through the cipher. If we have chosen Δ^*, ∇^* well, with good probability we obtain a difference of Δ^* halfway through the decryption of D, D', which lets us cover the remainder of the decryption with the backward differential $\Delta^* \to \Delta$. As a result, in a right quartet we will have the recognizable condition $Q \oplus Q' \in \Delta$. Many details have been omitted; a full description of the boomerang attack may be found in [17].

5.1 The middle twenty-four rounds

Consider a simplified 24-round Skipjack variant obtained by deleting four rounds from both ends of the real Skipjack cipher. This variant is intended to be relatively representative of a Skipjack cipher weakened to 24 rounds, in that it retains the symmetry between encryption and decryption.

Observe that there is a truncated differential of probability one through four A-rounds and eight B-rounds: $\Delta = (0, a, 0, 0) \xrightarrow{4r_A} (b, b, 0, 0) \xrightarrow{8r_B} (c, d, e, 0) = \Delta^*$. Due to the fact that the structure of an A-round is almost the inverse of the structure of a B-round, we also obtain a truncated differential of probability one for decryption through four B-rounds and eight A-rounds, specifically $\nabla = (f, 0, 0, 0) \xrightarrow{4r_B^{-1}} (g, g, 0, 0) \xrightarrow{8r_A^{-1}} (i, h, 0, j) = \nabla^*$. (Here a, b, \ldots, j can take on any non-zero value.) Finally, we use the backward differential $\Delta^* \to \Delta$ of probability 2^{-32} for decrypting through the first half of the cipher. This gives a success probability 2^{-32} of 2^{-16} of 2^{-1

To mount a boomerang attack first construct a plaintext pair P, P' with $P \oplus P' \in \Delta$. Denote the ciphertexts C, C'. Next obtain 2^{16} ciphertexts D by varying the first word in C, and in a similar manner obtain 2^{16} ciphertexts D'

⁷ Here the factor of 2^{-16} comes from the requirement that we get a difference of Δ^* halfway through the decryption of D, D', which happens when the fourth words of the two ∇^* differences are equal. In other words, if the ∇^* -difference is (i, h, 0, j) in the C, D pair and (i', h', 0, j') in the C', D' pair, we require that j = j' so that $(i, h, 0, j) \oplus (i', h', 0, j') \oplus (c, d, e, 0)$ will take the form of a Δ^* truncated difference. Finally, one must add a correction factor of $(1 - 2^{-16})^2$, because the differential $\Delta^* \to \Delta$ is not valid when $i \oplus i' \oplus c = 0$ or $h \oplus h' \oplus d = 0$.

by modifying C'. Note that the truncated differentials $\nabla \to \nabla^*$ for C, D and $\nabla \to \nabla^*$ for C', D' are simultaneously obeyed, so we get 2^{32} possible quartets, of which $2^{32} \times 2^{-16} (1-2^{-16})^2 \approx 2^{16}$ have a difference of the form Δ^* halfway through the decryptions of D, D'.

Each structure of 2^{17} texts contains a right quartet with probability $2^{16} \times 2^{-32} = 2^{-16}$. Right quartets can be recognized by the difference Δ in the plaintexts Q, Q'. This allows us to filter out all but 2^{-48} of the wrong quartets and after repeating the attack 2^{16} times, we expect to obtain one right quartet and one wrong quartet.

To reduce the number of texts required we can choose $2^{8.5}$ plaintexts by varying the second word and holding the first, third, and fourth words fixed; then for each of the $2^{8.5}$ resulting ciphertexts, we generate 2^{16} more variant ciphertexts and decrypt. Each pair of plaintexts then gives a structure, so 2^{16} structures can be obtained from this pack of $2^{8.5}$ plaintexts, and thus we expect to see the first right quartet after only $2^{24.5}$ chosen texts.

While we cannot recover key information from this 24-round version of Skip-jack using these techniques, we are able to distinguish this version from a random cipher with about $2^{8.5}-2^{9.5}$ chosen plaintexts and $2^{24.5}-2^{25.5}$ chosen ciphertexts. The same ideas can be applied to the inverse cipher to get a similar attack that uses $2^{24.5}-2^{25.5}$ chosen ciphertexts and $2^{8.5}-2^{9.5}$ chosen plaintexts.

5.2 The middle twenty-five rounds

Consider a Skipjack variant obtained by deleting the first three and last four rounds from the real Skipjack. We can use $2^{34.5}$ chosen texts to break 25 rounds of Skipjack with a $2^{61.5}$ work factor. One can use structures to bypass the first round subkey: vary the first and fourth words, and hold the middle two words fixed. With $2^{18.5}$ such plaintexts, one expects to find 2^{20} pairs of plaintexts which satisfy the desired relationship after the first round. After another $2^{34.5}$ chosen ciphertexts, one should find about 16 right quartets.

We then guess the first round subkey, and peel off the first round, checking for right quartets in the same way as in our 24-round attack. In this way, for each guess at the subkey we expect only about 16 of the wrong quartets to survive the filtering phase. This allows us to distinguish a right guess at the first round subkey from a wrong guess with good probability. In the former case 32 quartets will survive the filtering phase and in the latter only 16 quartets are expected to survive, which is a difference of four standard deviations. The analysis procedure can be performed with about $2^{34.5} \times 2^{32} = 2^{66.5}$ computations of G, which is a workload roughly equivalent to $2^{61.5}$ trial encryptions. In all, the attack recovers 32 key bits after $2^{61.5}$ work; the remaining 48 key bits can be found by trial decryption.

5.3 Comparison with miss-in-the-middle attacks

It is interesting to compare the complexity of boomerang attacks to Biham et al.'s miss-in-the-middle attacks [5] on the same reduced-round variants of Skipjack.

For 24 rounds, a boomerang attack needs $2^{24.5}$ – $2^{25.5}$ chosen texts to distinguish the cipher from a random permutation, whereas the miss-in-the-middle attack needs 2^{33} – 2^{35} chosen texts. For 25 rounds, our boomerang attack uses $2^{34.5}$ texts and $2^{61.5}$ work to recover the key, whereas the miss-in-the-middle attack [5] uses 2^{38} chosen texts and 2^{48} work. With regards to the data requirements, it appears that boomerang attacks compare favorably to miss-in-the-middle attacks for these reduced-round variants of Skipjack. However Biham et al. have demonstrated that miss-in-the-middle attacks can be used to analyze 31 rounds of Skipjack whereas boomerang attacks are currently restricted to no more than 25 rounds.

The boomerang attacks were aided by the fact that the (4A,8B) round structure (as found in the first half of the 24-round cipher) is weaker against truncated differentials in the encryption direction than in the decryption direction, while the (8A,4B) structure is weaker in the reverse direction. This property makes it easy to probe the (4A,8B,8A,4B) cipher from both ends at once with a boomerang attack. We suspect for similar reasons that a (16B,16A) structure might be easier to analyze with boomerang techniques than a (16A,16B) structure, which suggests that the ordering of the A-rounds and B-rounds may be quite important to the security of the Skipjack cipher.

6 Conclusion

In this paper we have described several interesting truncated differentials for Skipjack. These can be used in a variety of attacks, including particularly efficient attacks on reduced-round versions of Skipjack. The existence of such attacks and the effectiveness of truncated differentials demonstrates that Skipjack has unusual and surprising structural features. We also demonstrate the effectiveness of boomerang attacks on Skipjack. While they cannot be extended to attack 31 rounds of Skipjack like miss-in-the-middle attacks, for those reduced-round versions of Skipjack that can be compromised using both techniques, boomerang attacks are typically more effective than miss-in-the-middle attacks. We feel that attempts to extend existing boomerang attacks to more rounds could lead to more efficient attacks on Skipjack than are currently available. We leave it as a challenge to use our findings to find more efficient attacks on 16- to 31-round variants of Skipjack. Currently, an attack on the full 32 rounds of the cipher (other than by a brute force search for the key) remains elusive.

References

- E. Biham, A. Biryukov, O. Dunkelmann, E. Richardson, A. Shamir. Initial Observations on the Skipjack Encryption Algorithm. June 25, 1998. Available at http://www.cs.technion.ac.il/~biham/Reports/SkipJack/.
- 2. E. Biham, A. Biryukov, O. Dunkelmann, E. Richardson, and A. Shamir. Cryptanalysis of Skipjack-3XOR in 2^{20} time and using 2^9 chosen plaintexts. July 2, 1998. Available at
 - http://www.cs.technion.ac.il/~biham/Reports/SkipJack/.

- 3. E. Biham, A. Biryukov, O. Dunkelmann, E. Richardson, and A. Shamir. Cryptanalysis of Skipjack-4XOR. June 30, 1998. Available at
 - http://www.cs.technion.ac.il/~biham/Reports/SkipJack/.
- 4. E. Biham, A. Biryukov, and A. Shamir. Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR. SAC'98. Available at
 - http://www.cs.technion.ac.il/~biham/Reports/SkipJack/.
- 5. E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In J. Stern, editor, Advances in Cryptology Eurocrypt '99, volume 1592 of Lecture Notes in Computer Science, pages 12–23, 1999. Springer Verlag. Also available at
 - http://www.cs.technion.ac.il/~biham/Reports/SkipJack/.
- E. Biham and A. Shamir. Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag, New York, 1993.
- 7. L.R. Knudsen. Applications of higher order differentials and partial differentials. In B. Preneel, editor, Fast Software Encryption, volume 1008 of Lecture Notes in Computer Science, pages 196–211, 1995. Springer Verlag.
- 8. L.R. Knudsen, V. Rijmen, R.L. Rivest, and M.J.B. Robshaw. On the design and security of RC2. In S. Vaudenay, editor, *Fast Software Encryption*, volume 1372 of *Lecture Notes in Computer Science*, pages 206–219, 1998. Springer Verlag.
- E.F. Brickell, D.E. Denning, S.T. Kent, D.P. Maher, and W. Tuchman. Skipjack Review Interim Report. July 28, 1993.
- M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, Advances in Cryptology - Eurocrypt '93, volume 765 of Lecture Notes in Computer Science, pages 386–397, 1993. Springer Verlag.
- B. Schneier and D. Banisar. The Electronic Privacy Papers. John Wiley & Sons, 1997.
- 12. W. Diffie and S. Landau. Privacy on the Line. MIT Press, 1998.
- National Institute of Standards and Technology (NIST). FIPS Publication 186: Digital Signature Standard (DSS). May 1994.
- National Institute of Standards and Technology (NIST). FIPS Publication 180-1: Secure Hash Standard (SHS). May 1994.
- 15. National Security Agency. Skipjack and KEA algorithm specifications. May 1998. Available at http://csrc.ncsl.nist.gov/encryption/skipjack-1.pdf.
- 16. National Security Agency. NSA Releases Fortezza Algorithms, Press Release, June 24, 1998. Available at http://csrc.ncsl.nist.gov/encryption/nsa-press.pdf.
- D. Wagner. The boomerang attack. In L. Knudsen, editor, Fast Software Encryption, volume 1636 of Lecture Notes in Computer Science, pages 156–170, 1999. Springer Verlag.

Fast Correlation Attacks Based on Turbo Code Techniques

Thomas Johansson* and Fredrik Jönsson*

Dept. of Information Technology Lund University, P.O. Box 118, 221 00 Lund, Sweden {thomas,fredrikj}@it.lth.se

Abstract. This paper describes new methods for fast correlation attacks on stream ciphers, based on techniques used for constructing and decoding the by now famous turbo codes. The proposed algorithm consists of two parts, a preprocessing part and a decoding part. The preprocessing part identifies several parallel convolutional codes, embedded in the code generated by the LFSR, all sharing the same information bits. The decoding part then finds the correct information bits through an iterative decoding procedure. This provides the initial state of the LFSR.

Keywords. Stream ciphers, correlation attacks, convolutional codes, iterative decoding, turbo codes.

1 Introduction

Stream ciphers are generally faster than block ciphers in hardware, and have less complex hardware circuitry, implying a low power consumption. Furthermore, buffering is limited and in situations where transmission errors can occur the error propagation is limited. These are all properties that are especially important in, e.g., telecommunications applications.

Consider a binary additive stream cipher, i.e., a synchronous stream cipher in which the keystream, the plaintext, and the ciphertext are sequences of binary digits. The output sequence of the keystream generator, z_1, z_2, \ldots is added bitwise to the plaintext sequence m_1, m_2, \ldots , producing the ciphertext c_1, c_2, \ldots . The keystream generator is initialized through a secret key k, and hence, each key k will correspond to an output sequence. Since the key is shared between the transmitter and the receiver, the receiver can decrypt by adding the output of the keystream generator to the ciphertext, obtaining the message sequence, see Figure 1.

The design goal is to efficiently produce random-looking sequences that are as "indistinguishable" as possible from truly random sequences. Also, from a cryptanalysis point of view, a good stream cipher should be resistant against different kind of attacks, e.g., a known-plaintext attack. For a synchronous stream cipher, a known-plaintext attack is equivalent to the problem of finding the key

^{*} The authors are supported by the Foundation for Strategic Research - PCC under Grant 9706-09.

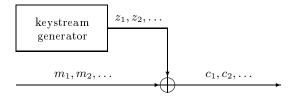


Fig. 1. Principle of binary additive stream ciphers

k that produced a given keystream z_1, z_2, \ldots, z_N . We hence assume that a given output sequence from the keystream generator z_1, z_2, \ldots, z_N is known to the cryptanalyst and that his task is to restore the secret key k.

In stream cipher design, it is common to use linear feedback shift registers, LFSRs, as building blocks in different ways. Furthermore, the secret key k is often chosen to be the initial state of the LFSRs.

Several classes of general cryptanalytic attacks against stream ciphers exist [13]. A very important, if not the most important, class of attacks on LFSR-based stream ciphers is correlation attacks. If one can, in some way, detect a correlation between the known output sequence and the output of one individual LFSR, it is possible to mount a "divide-and-conquer" attack on the individual LFSR [16,17,11,12]. Observe that there is no requirement of structure of any kind for the key generator. The only requirement is the fact that, if u_1, u_2, \ldots denotes the output of the particular LFSR, we have a correlation of the form $P(u_n = z_n) \neq 0.5$, $n \geq 1$. Other types of correlations may also apply.

A common methodology for producing random-like sequences from LFSRs is to combine the output of several LFSRs by a nonlinear Boolean function f with desired properties [13]. The purpose of f is to destroy the linearity of the LFSR sequences and hence provide the resulting sequence with a large linear complexity [13]. Note that for such a stream cipher, there is always a correlation between the output z_n and either one or a set output symbols from different LFSRs.

Finding a low complexity algorithm that successfully can use the existing correlation in order to determine a part of the secret key can be a very efficient way of attacking stream ciphers for which a correlation is identified. After the initializing ideas of Siegenthaler [16,17], Meier and Staffelbach [11,12] found a very interesting way to explore the correlation in what was called a fast correlation attack. A necessary condition is that the feedback polynomial of the LFSR has a very low weight. This work was followed by several papers, providing minor improvements to the initial results of Meier and Staffelbach, see [14,3,4,15]. However, the algorithms demonstrate true efficiency (good performance and low complexity) only if the feedback polynomial is of low weight. Based on these attacks, it is a general advise that the generator polynomial should not be of low weight when constructing stream ciphers.

More recently, a step in another direction was taken, and it was suggested to use convolutional codes in order to improve performance [8]. More precisely, it was shown that one can identify an embedded low-rate convolutional code in the code generated by the LFSR sequences. This convolutional code can then be decoded using, e.g., the Viterbi algorithm, and a correctly decoded information sequence will provide the correct initial state of the LFSR.

The purpose of this paper is to describe new algorithms for fast correlation attacks. They are based on combining the iterative decoding techniques introduced by Meier and Staffelbach [11,12] with the framework of embedded convolutional codes as suggested by the authors [8]. The proposed algorithm consists of two parts, a preprocessing part and a decoding part.

By considering permuted versions of the code generated by the LFSR sequences, several "parallel" embedded convolutional codes can be identified. They all share the same information sequence but have individual parity checks. This is the preprocessing part.

In the decoding part, the keystream z_1, z_2, \ldots, z_N is first used to construct sequences acting as received sequences for the above convolutional codes. These sequences are then used to find the correct information sequence by an iterative decoding procedure.

The code construction in the preprocessing part and the iterative decoding technique in the decoding part resemble very much the by now famous *turbo codes* and its decoding techniques [2]. Iterative decoding requires APP (a posteriori probability) decoding (also called MAP decoding), and for decoding convolutional codes one can use the famous BCJR algorithm [1], or preferably some modification of it [19,5].

For a fixed memory size, the proposed algorithm provides a better performance than previous methods. As a particular example taken from [14], consider a LFSR of length 40 and an observed sequence of length 40000 bits. Let 1-p be the correlation probability. Then for a certain memory size (B=13), the best known algorithm [8] is successful up to p=0.19, whereas the maximum p for the proposed algorithm lie in the range 0.20-0.27 when the number of parallel codes varies from one to 32. The price payed for increased performance is an increased computational complexity, but as argued in the paper, it is usually the available memory that limits the performance and not the computational complexity.

The paper is organized as follows. In Section 2 we give some preliminaries on the decoding model that is used for cryptanalysis, and in Section 3 and 4 we shortly review previous algorithms for fast correlation attacks. In Section 5 we present some of the new ideas in a basic algorithm using only one code. In Section 6 the use of several "parallel" codes is introduced and the complete algorithm is described. Simulation results are presented in Section 7. In Section 8 a parallelizable algorithm is proposed, and in Section 9 we conclude with some possible extensions.

2 Preliminaries

As most other authors [17,11,12,14,3], we use the approach of viewing our cryptanalysis problem as a decoding problem. An overview is given in Figure 2. Let the LFSR have length l and let the set of possible LFSR sequences be denoted by \mathcal{L} . Clearly, $|\mathcal{L}| = 2^l$ and for a fixed length N the truncated sequences from \mathcal{L} is also a linear [N, l] block code [10], referred to as \mathcal{C} . Furthermore, the keystream sequence $\mathbf{z} = z_1, z_2, \ldots, z_N$ is regarded as the received channel output and the LFSR sequence $\mathbf{u} = u_1, u_2, \ldots, u_N$ is regarded as a codeword from the linear block code \mathcal{C} .

Due to the correlation between u_n and z_n , we can describe each z_n as the output of the binary symmetric channel, BSC, when u_n was transmitted. The correlation probability 1-p, defined by $1-p=P(u_n=z_n)$, gives p as the crossover probability (error probability) in the BSC. W.l.o.g we assume p<0.5. This is all shown in Figure 2.

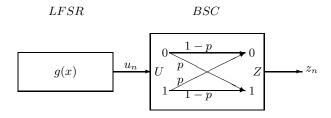


Fig. 2. Model for a correlation attack

The problem of cryptanalysis is now the following. Given the received word (z_1, z_2, \ldots, z_N) as output of the BSC(p), find the length N codeword from C that was transmitted.

It is known that the length N should be at least around $N_0 = l/(1 - h(p))$ for unique decoding [3], where h(p) is the binary entropy function. If the length N of the observed keystream sequence is small but allows unique decoding, the fastest methods for decoding are probabilistic decoding algorithms like Leon or Stern algorithms [9,18].

We assume instead received sequences of large length, $N >> N_0$. For this case, the fast correlation attacks [11,12] are applicable. These attacks resemble very much the iterative decoding process proposed by Gallager [6] for low-weight parity-check codes. The drawback is the fact that the above attacks require the feedback polynomial g(x) (or any multiple of g(x) of modest degree) to have a low weight. Hence one usually refrain from using such feedback polynomials in stream cipher design.

3 Fast Correlation Attacks - An Overview

In [11,12] Meier and Staffelbach presented two algorithms, referred to as A and B, for fast correlation attacks. Instead of the exhaustive search as originally suggested in [17], the algorithms are based on using certain parity check equations created from the feedback polynomial of the LFSR.

In the first pass, a set of suitable parity check equations in the code \mathcal{C} is found. The second pass uses these parity check equations in a fast decoding algorithm to recover the transmitted codeword and hence the initial state of the LFSR.

Parity check equations in [11,12] were created in two separate steps. Let $g(x) = 1 + g_1x^1 + g_2x^2 + \ldots + g_lx^l$ be the feedback polynomial, and t the number of taps of the LFSR, i.e., the weight of g(x) (the number of nonzero coefficients) is t+1. Symbol number n of the LFSR sequence, u_n , can then be written as $u_n = g_1u_{n-1} + g_2u_{n-2} + \ldots + g_lu_{n-l}$. Since the weight of g(x) is t+1, there are the same number of relations involving a fixed position u_n . Hence, we get in this way t+1 different parity check equations for u_n . Secondly, using the fact that $g(x)^j = g(x^j)$ for $j=2^i$, parity check equations are also generated by repeatedly squaring the polynomial g(x). The obtained parity check equations are then (essentially) valid in each index position of \mathbf{u} .

The number of parity check equations, denoted m, that can be found in this way is $m \approx \log(\frac{N}{2l})(t+1)$, where log uses base 2 [11,12].

In the second pass, we have m equations for position u_n as,

$$u_n + b_1 = 0,$$

 $u_n + b_2 = 0,$
 \vdots
 $u_n + b_m = 0,$
(1)

where each b_i is the sum of t different positions of \mathbf{u} . Applying the above relations to the keystream we can introduce L_i , $\leq i \leq m$, defined as the following sums,

$$z_n + y_1 = L_1$$

$$z_n + y_2 = L_2$$

$$\vdots$$

$$z_n + y_m = L_m.$$
(2)

where y_i is the sum of the positions in the keystream **z** corresponding to the positions in b_i . Assume that h out of the m equations in (1) hold, i.e.,

$$h = |\{i : L_i = 0, 1 \le i \le m\}|,$$

The probability $p^* = P(u_n = z_n | h \text{ equations hold})$ is calculated as

$$p^* = \frac{ps^h(1-s)^{m-h}}{ps^h(1-s)^{m-h} + (1-p)(1-s)^h s^{m-h}},$$

where $p = P(u_n = z_n)$, and $s = P(b_i = y_i)$. This is used as an estimate of whether z_n was correct or not.

Two different decoding methods was suggested in [11,12]. The first algorithm, called Algorithm A, can shortly be described as follows: Calculate the probabilities p^* for each bit in the keystream, select the l positions with highest value of p^* , and calculate a candidate initial state. Finally, find the correct value by checking the correlation between the sequence and the keystream for different small modifications of the candidate initial state.

The second algorithm, called Algorithm B, uses instead an iterative approach. The algorithm uses two parameters p_{thr} and N_{thr} .

- 1. For all symbols in the keystream, calculate p^* and determine the number of positions N_w with $p^* < p_{thr}$.
- 2. If $N_w < N_{thr}$ repeat step 1 with p replaced by p^* in each position.
- 3. Complement the bits with $p^* < p_{thr}$ and reset the probabilities to p.
- 4. If not all equations are satisfied go to step 1.

This iterative approach is fundamental for our considerations, and we refer to [11,12] for more details.

The algorithms above work well when the LFSR contains few taps, but for LFSRs with many taps the algorithms fail. The reason for this failure is that for LFSRs with many taps each parity check equation gives a very small average correction and hence many more equations are needed in order to succeed. Or in other words, the maximum correlation probability p that the algorithms can handle is much lower if the LFSR has many taps ($\approx l/2$). Minor improvements were suggested in, e.g., [14] and [3].

4 Fast Correlation Attacks Based on Convolutional Codes

The general idea behind this algorithm, proposed recently in [8], is to consider slightly more advanced decoding algorithms including memory, but which still have a low decoding complexity. This allows weaker restrictions on the parity check equations that can be used, leading to many more and more powerful equations. The work in [8] then uses the Viterbi algorithm as its decoding algorithm in the decoding part.

We now review the basic results of [8]. The algorithm transforms a part of the code C stemming from the LFSR sequences into a convolutional code. The encoder of this convolutional code is created by finding suitable parity check equations from C. It is assumed that the reader is familiar with basic concepts regarding convolutional codes (see also [8]).

Let B be a fixed memory size and let R denote the rate. In a convolutional encoder with memory B and rate R = 1/(m+1) the vector \mathbf{v}_n of codeword symbols at time n,

$$\mathbf{v}_n = (v_n^{(0)}, v_n^{(1)}, \dots, v_n^{(m)}),$$

is of the form

$$\mathbf{v}_n = u_n \mathbf{g}_0 + u_{n-1} \mathbf{g}_1 + \dots u_{n-B} \mathbf{g}_B, \tag{3}$$

where each \mathbf{g}_i is a vector of length (m+1). The task in the first pass of the algorithm is to find suitable parity check equations that will determine the vectors $\mathbf{g}_i, 0 \leq i \leq m$, defining the convolutional code.

Let us start with the linear code C stemming from the LFSR sequences. There is a corresponding $l \times N$ generator matrix G_{LFSR} , such that $\mathbf{u} = \mathbf{u}_0 G_{LFSR}$, where \mathbf{u}_0 is the initial state of the LFSR. The generator matrix is furthermore written in systematic form, i.e., $G_{LFSR} = (I_l \ Z)$, where I_l is the $l \times l$ identity matrix.

We are now interested in finding parity check equations that involve a current symbol u_n , and an arbitrary linear combination of the B previous symbols u_{n-1}, \ldots, u_{n-B} , together with at most t other symbols. Clearly, t should be small and in this description t = 2 is mainly considered.

To find these equations, start by considering the index position n = B + 1. Introduce the following notation for the generator matrix,

$$G_{LFSR} = \begin{pmatrix} I_{B+1} & Z_{B+1} \\ 0_{l-B-1} & Z_{l-B-1} \end{pmatrix}. \tag{4}$$

Parity check equations for u_{B+1} with weight t outside the first B+1 positions can then be found by finding linear combinations of t columns of Z_{l-B-1} that adds to the all zero vector.

For the case t=2 the parity check equations can be found in a very simple way as follows. A parity check equation with t=2 is found if two columns from G_{LFSR} have the same value when restricted to the last l-B-1 entries (the Z_{l-B-1} part). Hence, we simply put each column of Z_{l-B-1} into one of 2^{l-B-1} different "buckets", sorted according to the value of the last l-B-1 entries. Each pair of columns in each bucket will provide us with one parity check equation, provided u_{B+1} is included.

Assume that the above procedure gives us a set of m parity check equations for u_{B+1} , written as

$$u_{B+1} + \sum_{i=1}^{B} c_{i1} u_{B+1-i} + \sum_{i=1}^{\leq t} u_{j_{i1}} = 0,$$

$$u_{B+1} + \sum_{i=1}^{B} c_{i2} u_{B+1-i} + \sum_{i=1}^{\leq t} u_{j_{i2}} = 0,$$

$$\vdots$$

$$u_{B+1} + \sum_{i=1}^{B} c_{im} u_{B+1-i} + \sum_{i=1}^{\leq t} u_{j_{im}} = 0.$$
(5)

It directly follows from the cyclic structure of the LFSR sequences that the same set of parity checks is valid for any index position n simply by shifting all the symbols in time, resulting in

$$u_{n} + \sum_{i=1}^{B} c_{i1} u_{n-i} + b_{1} = 0,$$

$$u_{n} + \sum_{i=1}^{B} c_{i2} u_{n-i} + b_{2} = 0,$$

$$\vdots$$

$$u_{n} + \sum_{i=1}^{B} c_{im} u_{n-i} + b_{m} = 0,$$
(6)

where $b_k = \sum_{i=1}^{\leq t} u_{j_{ik}}$, $1 \leq k \leq m$ is the sum of (at most) t positions in \mathbf{u} . Using the equations above one can create an $R = \frac{1}{m+1}$ bi-infinite systematic convolutional encoder. Recall that the generator matrix for such a code is of the form

$$G = \begin{pmatrix} \ddots & \ddots & \ddots & & & & \\ & \mathbf{g_0} & \mathbf{g_1} & \cdots & \mathbf{g_B} & & & & \\ & & \mathbf{g_0} & \mathbf{g_1} & \cdots & \mathbf{g_B} & & & & \\ & & & \ddots & & \ddots & & & \end{pmatrix}, \tag{7}$$

where the blank parts are regarded as zeros. Identifying the parity check equations from (6) with the description form of the convolutional code as in (7) gives

$$\begin{pmatrix} \mathbf{g}_{0} \\ \mathbf{g}_{1} \\ \vdots \\ \mathbf{g}_{B} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & c_{11} & c_{12} & \dots & c_{1m} \\ 0 & c_{21} & c_{22} & \dots & c_{2m} \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & c_{B1} & c_{B2} & \dots & c_{Bm} \end{pmatrix}.$$
(8)

For each defined codeword symbol $v_n^{(i)}$ in the convolutional code one has an estimate of that symbol from the transmitted sequence **z**. If $v_n^{(i)} = u_n$ (an information bit) then $P(v_n^{(i)} = z_n) = 1 - p$. Otherwise, if $v_n^{(i)} = u_{j_{1i}} + u_{j_{2i}}$ from (6) then $P(v_n^{(i)} = z_{j_{1i}} + z_{j_{2i}}) = (1 - p)^2 + p^2$. Using these estimates one can construct a sequence

$$\mathbf{r} = \dots r_n^{(0)} r_n^{(1)} \dots r_n^{(m)} r_{n+1}^{(0)} r_{n+1}^{(1)} \dots r_{n+1}^{(m)} \dots,$$

where $r_n^{(0)} = z_n$ and $r_n^{(i)} = z_{j_{1i}} + z_{j_{2i}}$, $1 \le i \le m$, that plays the role of a received sequence for the convolutional code, where $P(v_n^{(0)} = r_n^{(0)}) = 1 - p$ and $P(v_n^{(i)} = r_n^{(i)}) = (1-p)^2 + p^2 \text{ for } 1 \le i \le m.$

To recover the initial state of the LFSR it is enough to decode l consecutive information bits correctly. Optimal decoding (ML decoding) of the convolutional code using the Viterbi algorithm can thus be performed.

The original Viterbi algorithm assumes that the convolutional encoder starts in state 0. However, in this application there is neither a starting state, nor an ending state for the trellis corresponding to the convolutional code. Hence, one runs the Viterbi algorithm over a number of "dummy" information symbols, placed before and after the l information symbols that one tries to decode correctly, see [8]. A suitable choice is to decode over J = l + 10B information symbols, where the l symbols in the middle are regarded as the l information bits that we want to estimate. The particular choice of J is based on heuristics for the decision distance of the decoding algorithm.

5 Some New Ideas for Fast Correlation Attacks

Two methods for decoding a noisy LFSR sequence have been described in Section 3 and 4. The Meier and Staffelbach Algorithm B calculates an a posteriori probability for each symbol of the complete received sequence and then iteratively tries to improve these probabilities by recalculating them. The procedure is based on very simple (memoryless) parity checks. The method of Section 4 uses instead convolutional codes but uses a simple Viterbi decoding procedure on a small part of the received sequence.

The ideas to be proposed try to combine the best parts of both methods into a single algorithm. The first and basic construction uses one convolutional code (Section 4 method) and then applies an APP (a posteriori probability) decoding algorithm in order to provide an a posteriori probability for each symbol in a certain part of the received sequence. Optimal APP decoding (also referred to as MAP decoding) on a convolutional code can be performed by the famous BCJR algorithm [1], or variations of it. The a posteriori probabilities are then fed back as a priori probabilities and in this fashion the procedure is iterated until convergence. This is now described in more detail.

The first step involves computing parity check equations for a convolutional code with fixed memory B. We follow the procedure of Section 4 and compute all parity check equations with t=2 involving the particular index position B+1, as given by (5). Parity checks for index position B+1+i are then immediately obtained through a cyclic shift of the original parity checks with i steps, as in (6). We refer to Section 4 for a review of the details. Write the obtained parity check equations in the form

$$u_{n} + \sum_{i=1}^{B} c_{i1} u_{n-i} + u_{i_{n1}} + u_{j_{n1}} = 0,$$

$$u_{n} + \sum_{i=1}^{B} c_{i2} u_{n-i} + u_{i_{n2}} + u_{j_{n2}} = 0,$$

$$\vdots$$

$$u_{n} + \sum_{i=1}^{B} c_{im} u_{n-i} + u_{i_{nm}} + u_{j_{nm}} = 0.$$
(9)

The convolutional code is defined by all codeword sequences \mathbf{v} ,

$$\mathbf{v} = \dots v_n^{(0)} v_n^{(1)} \dots v_n^{(m)} v_{n+1}^{(0)} v_{n+1}^{(1)} \dots v_{n+1}^{(m)} \dots, \qquad B+1 \le n \le J,$$

where

$$v_n^{(0)} = u_n, \quad v_n^{(k)} = u_n + \sum_{i=1}^B c_{ik} u_{n-i}, \quad 1 \le k \le m.$$

Observe that the code is defined only over the interval $B+1 \leq n \leq J$. Since there are neither a starting state nor an ending state for the code trellis, we again decode over J-B>l information symbols. Following Section 4, we choose J=10B+l as a starting point. Through simulations one can later adjust J to the most suitable value. Furthermore, the starting state, denoted $\mathbf{s}_{\mathbf{s}}$, of the trellis (start value for the memory contents of the convolutional code) is given a probability distribution which is $P(\mathbf{s}_{\mathbf{s}})=P(u_1,u_2,\ldots,u_B)$.

The second step is the APP decoding phase. After receiving a sequence \mathbf{z} , construct a sequence \mathbf{r} acting as a received sequence for the convolutional code by

$$\mathbf{r} = \dots r_n^{(0)} r_n^{(1)} \dots r_n^{(m)} r_{n+1}^{(0)} r_{n+1}^{(1)} \dots r_{n+1}^{(m)} \dots, \qquad B+1 \le n \le J,$$

where

$$r_n^{(0)} = z_n, \quad r_n^{(k)} = z_{i_{nk}} + z_{j_{nk}}, \quad 1 \le k \le m.$$

Next, we transfer the a priori probabilities of ${\bf u}$ to the sequence ${\bf v}$ by

$$P(v_n^{(0)} = r_n^{(0)}) = P(u_n = z_n), \tag{10}$$

where $P(u_n = z_n) = 1 - p$ only in the first iteration, and

$$P(v_n^{(k)} = r_n^{(k)}) = (1 - p)^2 + p^2, 1 \le k \le m.$$
(11)

Then decode the constructed sequence **r** stemming from a codeword **v** of the convolutional code using an APP decoding algorithm. The original BCJR algorithm requires storage of the whole trellis. However, suboptimal versions of the BCJR algorithm, see [19,5], remove this problem with a negligible decrease in performance. This procedure provides us with the a posteriori probabilities for the information sequence, i.e.,

$$P(v_{B+1}^{(0)}|\mathbf{r}), P(v_{B+2}^{(0)}|\mathbf{r}), \dots, P(v_J^{(0)}|\mathbf{r}).$$

Finally, since $v_{B+1}^{(0)} = u_{B+1}, v_{B+2}^{(0)} = u_{B+2}, \ldots$ this information is fed back as new a priori probabilities for $(u_{B+1}, u_{B+2}, \ldots, u_J)$ and the a priori probabilities of the codeword sequence \mathbf{v} of the convolutional code is recalculated. The decoding procedure is performed a second time, and this procedure is iterated 2-5 times (until convergence).

Basic algorithm description:

Input: The $l \times N$ generator matrix G_{LFSR} for the code generated by a LFSR; the received sequence \mathbf{z} ; the error probability p; the number of iterations I.

- 1. (Precomputation) For each position n, $B+1 \le n \le J$, in G_{LFSR} , find the set of parity check equations of the form (9) and construct the convolutional code.
- 2. (Decoding phase) After receiving \mathbf{z} , construct the a priori probability vector $(P(u_{B+1}), P(u_{B+2}), \ldots, P(u_J))$ by $P(u_n = z_n) = 1 p$. Construct the received sequence \mathbf{r} by

$$r_n^{(0)} = z_n, \quad r_n^{(k)} = z_{i_{nk}} + z_{j_{nk}}, \quad 1 \le k \le m.$$

and the corresponding a priori probabilities for \mathbf{v}_n , $B+1 \leq n \leq J$ by

$$P(v_n^{(k)} = r_n^{(k)}) = (1-p)^2 + p^2, 1 \le k \le m.$$

3. (Decoding phase) Update

$$P(v_n^{(0)}) = P(u_n), \quad B+1 \le n \le J.$$

Run the APP decoding algorithm with starting state distribution $P(\mathbf{s_s}) = P(u_1, u_2, \dots, u_B)$. Receive the a posteriori probabilities

$$P(v_{B+1}^{(0)}|\mathbf{r}), P(v_{B+2}^{(0)}|\mathbf{r}), \dots, P(v_J^{(0)}|\mathbf{r}).$$

Since $v_n^{(0)} = u_n$, set

$$P(u_{B+1}) \leftarrow P(v_{B+1}^{(0)}|\mathbf{r}), P(u_{B+2}) \leftarrow P(v_{B+2}^{(0)}|\mathbf{r}), \dots, P(u_J) \leftarrow P(v_J^{(0)}|\mathbf{r}).$$

4. If the number of iterations < I go to 3., otherwise select the most probable value for each of the symbols $u_{5B+1}, u_{5B+2}, \ldots, u_{5B+l}$, calculate the initial state \mathbf{u}_0 and check if it is correct.

We end by presenting some simulation results for the basic algorithm. The obtained results are compared with the results in [11,12,14,8]. We choose to use the same case as tabulated in [14,8], which is based on a LFSR with length l=40, and a weight 17 feedback polynomial.

Ī	[11,12], Alg B.	[14]	[8]			Basic algorithm		
			B = 13	B = 14	B = 15	B = 13	B = 14	B = 15
\prod	0.092	0.096	0.19	0.22	0.26	0.20	0.23	0.26

Table 1. Maximum p for some different algorithms when N=40000 and B=13,14,15.

6 Algorithms Based on Turbo Code Techniques

One of the most revolutionary ideas in coding theory the last decade has been the introduction of turbo codes. The original turbo code [2] consists of two convolutional codes, where the information bits are directly fed into one of them and an interleaved version of the same information bits are fed into the other convolutional code. The fundamentally new idea was the proposed decoding scheme, which uses an iterative procedure. Decode the first code using an APP decoding algorithm which provides a posteriori probabilities for all information symbols. Use these as a priori information when decoding the second code using again APP decoding. The obtained a posteriori probabilities are now used as a priori information when decoding the first code a second time, and the procedure continues in this iterative fashion.

Much the same ideas as described above can be applied to our decoding problem. Instead of using just one fixed convolutional code, as in the basic algorithm described in Section 5, we will show how to find and use two or more different convolutional codes. These are obtained by randomly permuting the index positions of the original code in the interval B+1...J.

The a posteriori probability vector, which is the result of the decoding phase, from one code is fed as a priori information to the other code. This is viewed in Figure 3. A problem arises, however, since we need parity check equations for

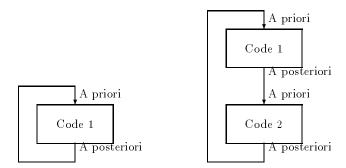


Fig. 3. The basic algorithm and the turbo code algorithm with two constituent codes.

permuted versions of the code \mathcal{C} . The shifting technique will no longer provide this for all indices, since after the column permutation the new code has no longer the cyclic properties of \mathcal{C} . To overcome this problem we simply search for all valid parity check equations in each index position. It will increase the precomputation time by a factor J-B, but for the case t=2 this is not at all a problem. Hence, this procedure will create different parity check equations for different index positions, thus leading to a timevarying convolutional code (in opposite to the code in Section 5). Also the number of parity checks will vary with n.

In order to find the parity check equations for an index position n, where $B+1 \leq n \leq J$, write the permuted generator matrix in the form

$$G_{LFSR} = \begin{pmatrix} Z_{11} & I_{B+1} & Z_{12} \\ Z_{21} & 0_{l-B-1} & Z_{22} \end{pmatrix}, \tag{12}$$

where Z_{21} has length n-B-1 and Z_{22} has length N-n. Then put each column of Z_{22} together with its index position into one of 2^{l-B-1} different "buckets", sorted according to the column value. Each pair of columns in each bucket will provide us with one valid parity check equation (of the form (9)) for index position n, provided u_n is included. Finally, since the number of parity checks will vary with

n, we introduce m(n) as the number of found parity checks for index position n. The parity check equations for index position n is written as

$$u_{n} + \sum_{i=1}^{B} c_{i1}u_{n-i} + u_{i_{n1}} + u_{j_{n1}} = 0,$$

$$u_{n} + \sum_{i=1}^{B} c_{i2}u_{n-i} + u_{i_{n2}} + u_{j_{n2}} = 0,$$

$$\vdots$$

$$u_{n} + \sum_{i=1}^{B} c_{im(n)}u_{n-i} + u_{i_{nm(n)}} + u_{j_{nm(n)}} = 0.$$
(13)

For each n, the constants defining the parity check equations,

$$\begin{pmatrix} \mathbf{g}_{0}(n) \\ \mathbf{g}_{1}(n) \\ \vdots \\ \mathbf{g}_{B}(n) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & c_{11} & c_{12} & \dots & c_{1m(n)} \\ 0 & c_{21} & c_{22} & \dots & c_{2m(n)} \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & c_{B1} & c_{B2} & \dots & c_{Bm(n)} \end{pmatrix}.$$
(14)

must be stored in order to build the trellis in the decoding phase. After this precomputation phase, the decoding of the first code follows the procedure of Section 5, but the resulting a posteriori probabilities are now fed as a priori information to the second decoder. The same procedure is repeated until all decoders have completed their task, and then the resulting a posteriori information is fed back to the first decoder, starting the second iteration. After 2-5 iterations the decoding phase is completed. A comprehensive description of the procedure for M constituent codes/decoders follows.

Turbo algorithm description:

Input: The $l \times N$ generator matrix G_{LFSR} for the code generated by the LFSR; the received sequence \mathbf{z} ; the error probability p; the number of iterations I; the number of constituent codes M.

- 1. (Precomputation) Let π_2, \ldots, π_M be M-1 random permutations permuting indices $B+1,\ldots,J$ and leaving the other indices fixed. Let $G_1=G_{LFSR}, G_2=\pi_2(G_{LFSR}),\ldots, G_M=\pi_M(G_{LFSR})$ be generator matrices for M different codes which are all permuted versions of G_{LFSR} . For G_i , let $\pi_i(\mathbf{z})$ be the received sequence, $2 \leq i \leq M$. Then find all parity checks of the form (13) for each G_i , $1 \leq i \leq M$. Initiate $i \leftarrow 1$.
- 2. (Decoding phase) After receiving \mathbf{z} , construct the a priori probability vector $(P(u_{B+1}), P(u_{B+2}), \dots, P(u_J))$ by $P(u_n = z_n) = 1 p$. For each G_i , construct the received sequence \mathbf{r} by

$$r_n^{(0)} = z_n, \quad r_n^{(k)} = z_{i_{nk}} + z_{j_{nk}}, \quad 1 \le k \le m.$$

and the corresponding a priori probabilities for \mathbf{v}_n , $B+1 \leq n \leq J$ by

$$P(v_n^{(k)} = r_n^{(k)}) = (1-p)^2 + p^2, 1 \le k \le m.$$

3. (Decoding phase) For G_i , update

$$P(v_n^{(0)}) = P(u_n), \quad B+1 \le n \le J.$$

Run the MAP algorithm on G_i with starting state distribution $P(\mathbf{s_s}) = P(u_1, u_2, \dots u_B)$. Receive the a posteriori probabilities

$$P(v_{B+1}^{(0)}|\mathbf{r}), P(v_{B+2}^{(0)}|\mathbf{r}), \dots, P(v_J^{(0)}|\mathbf{r}).$$

Set

$$P(u_{B+1}) \leftarrow P(v_{B+1}^{(0)}|\mathbf{r}), P(u_{B+2}) \leftarrow P(v_{B+2}^{(0)}|\mathbf{r}), \dots, P(u_J) \leftarrow P(v_J^{(0)}|\mathbf{r}).$$

Let $i \leftarrow i+1$ and if i=M+1 then $i \leftarrow 1$.

4. If the number of iterations $< I \cdot M$ go to 3., otherwise select the most probable value for each of the symbols $u_{5B+1}, u_{5B+2}, \ldots, u_{5B+l}$, calculate the initial state \mathbf{u}_0 and check it for correctness.

7 Performance of the Turbo Algorithm

In this section we present some simulation results for the turbo algorithm. The parameter values are J=10B+l and I=3. In Table 2 we show the maximum error probability for a received sequence of length N=40000 when the memory B is varying in the range 10-13 and the number of constituent codes is 1,2,4,8 and 16. Table 3 then shows the same for length N=400000.

B	[8]	M = 1	M = 2	M = 4	M = 8	M = 16
12	0.12	0.18	0.21	0.22	0.23	0.25
13	0.19	0.20	0.22	0.24	0.25	0.26
14	0.22	0.23	0.24	0.26	0.27	0.28
15	0.26	0.26	0.27	0.29	0.30	0.30

Table 2. Maximum p for turbo algorithm with B = 12, ..., 15 and varying M when N = 40000.

We can see the performance improvement with growing M for fixed B. A few comments regarding computational complexity and memory requirements are in place.

If one uses the suboptimal APP decoding algorithm in [19] the memory requirements will be roughly the same as in Viterbi decoding. The computational complexity for the algorithm in [19] is roughly a factor 3 higher compared to the Viterbi algorithm, since it runs through the trellis three times. There are also slightly different operations performed in the algorithms. The computational complexity is then further increased a factor M when the turbo algorithm with

B	[8]	M = 1	M=2	M = 4	M = 8	M = 16
10	0.31	0.31	0.33	0.34	0.35	0.36
11	0.34	0.34	0.36	0.37	0.38	0.38
12	0.36	0.37	0.38	0.38	0.39	0.39
13	0.37	0.39	0.40	0.40	0.41	0.41

Table 3. Maximum p for turbo algorithm with B = 10, ..., 13 and varying M when N = 400000.

M constituent codes are considered. Finally, we iterate at least twice. To conclude, for fixed parameters B and N, the turbo algorithm have roughly the same memory requirements, but an increase of computational complexity of at least a factor 6M.

It is important to note that in many cases, the possible performance is not limited by the computational complexity, but rather, limited by the required memory. For example, if N=40000, the maximal memory size that our current implementation could handle for the basic algorithm on a regular PC (for the example given in Table 2) was B=17, but this required only roughly half an hour CPU time. Hence, in this case we do not consider the penalty of increased computational complexity to be severe.

8 A Parallel Version of the Turbo Algorithm

As can be seen from the description of the turbo algorithm, it is not directly parallelizable (the APP decoding can be partly parallelized). Since it is very reasonable to assume that the opponent can compute in parallel, we shortly describe a modified turbo algorithm. Assume that the opponent has access to M different processors. He then constructs M different constituent convolutional codes exactly as described in Section 6 using some suitable memory B. After having received the keystream \mathbf{z} , the received sequences \mathbf{r} are constructed and the a priori probabilities are calculated. Next, processor number i works as the APP decoder for code number i, with the a priori probabilities as input. Observe that all the decoders work on the same input. Each decoder then outputs an a posteriori probability vector.

For each position n, an overall a posteriori probability for that position, $P(u_n = z_n)$ is calculated. Let $P_i(u_n = z_n | \mathbf{r})$ be the a posteriori probability stemming from code i. Then the overall a posteriori probability for this algorithm is given by

$$P(u_n = z_n) = \frac{\prod_{i=1}^{M} P_i(u_n = z_n | \mathbf{r})}{\prod_{i=1}^{M} P_i(u_n = z_n | \mathbf{r}) + \prod_{i=1}^{M} (1 - P_i(u_n = z_n | \mathbf{r}))}.$$
 (15)

The probability vector is then fed back as a priori information, and the same process is repeated again. The structure is depicted in Figure 4. The performance is slightly worse than the turbo algorithm. Some simulation result for N=40000 are given in Table 4.

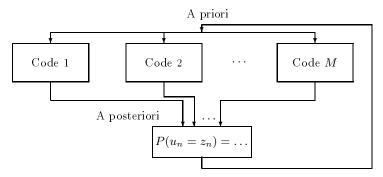


Fig. 4. The parallel turbo code algorithm combining the probabilities as in (15).

	M = 1	M=2	M = 4	M = 8	M = 16
					0.26
Parallel turbo	0.20	0.22	0.23	0.23	0.24

Table 4. Maximum p for the parallel turbo algorithm when N=40000 and B=13.

9 Conclusions

In this work we have shown how iterative decoding techniques based on the ideas from the construction and decoding of turbo codes can be used as a basis for correlation attacks on stream ciphers. The performance has been demonstrated through simulations. Still, many possible variations of the proposed type of fast correlation attacks exist, that need to be examined. The proposed iterative decoding techniques have opened for other possibilities that can be considered in the future. We mention two possible extensions.

- Reduced complexity decoding. As noted in this work, the main performance limit for the proposed decoding algorithms as well as for [8] is the memory requirement. A possible way to overcome this problem is to consider suboptimal decoding algorithms with reduced memory. Examples of such are list decoding and different sequential decoding algorithms [7].
- Other iterative decoding structures. An alternative to the decoding structure in this work, as given in Figure 3, could be the following. Consider all index positions that are used to build parity check equations for the convolutional code. Now consider these positions as information symbols for another convolutional code, and find parity checks for this code. Decoding this code will provide APP probabilities for its information symbols and hence more reliable parity checks for the first code. This idea is easily generalized to more complex decoding structures.

References

- L. R. Bahl, J. Cooke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, 1974, pp. 284-287.
- C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit errorcorrecting coding and decoding," Proc., IEEE Int. Conf. on Communications, ICC'93, 1993, pp. 1064-1070.
- 3. V. Chepyzhov, and B. Smeets, "On a fast correlation attack on certain stream ciphers", In Advances in Cryptology-EUROCRYPT'91, Lecture Notes in Computer Science, vol. 547, Springer-Verlag, 1991, pp. 176-185.
- A. Clark, J. Golic, E. Dawson, "A comparison of fast correlation attacks", Fast Software Encryption, FSE'96, Lecture Notes in Computer Science, Springer-Verlag, vol. 1039, 1996, pp. 145-158.
- J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-42, 1996, pp. 429-445.
- R. G. Gallager, Low-Density Parity-Check Codes, MIT Press, Cambridge, MA, 1963.
- R. Johanesson, K. Sh. Zigangirov, Fundamentals of convolutional coding, IEEE Press, New York, 1999.
- 8. T. Johansson, F. Jönsson, "Improved fast correlation attacks on stream ciphers via convolutional codes", Advances in Cryptology-EUROCRYPT'99, Lecture Notes in Computer Science, vol. 1592, Springer-Verlag, 1999, pp. 347-362.
- 9. J. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes", *IEEE Trans. Information Theory*, vol. 34, 1988, pp. 1354-1359.
- F. MacWilliams, N. Sloane, The theory of error correcting codes, North Holland, 1977.
- W. Meier, and O. Staffelbach, "Fast correlation attacks on stream ciphers", Advances in Cryptology-EUROCRYPT'88, Lecture Notes in Computer Science, vol. 330, Springer-Verlag, 1988, pp. 301-314.
- W. Meier, and O. Staffelbach, "Fast correlation attacks on certain stream ciphers", *Journal of Cryptology*, vol. 1, 1989, pp. 159-176.
- A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.
- M. Mihaljevic, and J. Golic, "A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence", Advances in Cryptology-AUSCRYPT'90, Lecture Notes in Computer Science, vol. 453, Springer-Verlag, 1990, pp. 165-175.
- 15. W. Penzhorn, "Correlation attacks on stream ciphers: Computing low weight parity checks based on error correcting codes", Fast Software Encryption, FSE'96, Lecture Notes in Computer Science, vol. 1039, Springer-Verlag, 1996, pp. 159-172.
- T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications", *IEEE Trans. on Information Theory*, vol. IT-30, 1984, pp. 776-780.
- 17. T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only", *IEEE Trans. on Computers*, vol. C-34, 1985, pp. 81-85.
- 18. J. Stern, "A method for finding codewords of small weight," Coding Theory and Applications, Springer-Verlag, 1989, pp. 106-113.
- 19. A. Trofimov, K. Zigangirov, "A posteriori probability decoding of convolutional codes", to appear in *Problems of Information Transmission*, 1999.

Highly Nonlinear Resilient Functions Optimizing Siegenthaler's Inequality

Subhamoy Maitra¹ and Palash Sarkar²

Computer and Statistical Service Center, Indian Statistical Institute 203, B T Road, Calcutta 700 035, India subho@isical.ac.in
Applied Statistics Unit, Indian Statistical Institute 203, B T Road, Calcutta 700 035, India palash@isical.ac.in

Abstract. Siegenthaler proved that an n input 1 output, m-resilient (balanced mth order correlation immune) Boolean function with algebraic degree d satisfies the inequality : $m+d \leq n-1$. We provide a new construction method using a small set of recursive operations for a large class of highly nonlinear, resilient Boolean functions optimizing Siegenthaler's inequality m+d=n-1. Comparisons to previous constructions show that better nonlinearity can be obtained by our method. In particular, we show that as n increases, for almost all m, the nonlinearity obtained by our method is better than that provided by Seberry et al in Eurocrypt'93. For small values of n, the functions constructed by our method is better than or at least comparable to those constructed using the methods provided in papers by Filiol et al and Millan et al in Eurocrypt'98. Our technique can be used to construct functions on large number of input variables with simple hardware implementation.

Keywords: Stream Cipher, Boolean Function, Algebraic Degree, Correlation Immunity, Nonlinearity, Balancedness.

1 Introduction

In stream cipher cryptography, the message is considered to be a stream of bits. The cipher is obtained by bitwise XORing (addition over GF(2)) the message with a sequence of bits called the key stream. In most common models of stream ciphers the key stream is produced by using a Boolean function to combine the output sequences of several Linear Feedback Shift Registers (LFSRs). If the combining Boolean function is not properly chosen, then the system becomes susceptible to several kinds of cryptanalytic attacks. An important class of divide-and-conquer attacks on such systems was proposed by Siegenthaler [18]. Moreover, Siegenthaler [17] himself introduced a class of Boolean functions, the set of correlation immune functions, which can resist such attacks. However, it is not sufficient to use functions with only correlation immunity, since certain types of correlation immune functions are susceptible to other kinds of attacks. For example, it is well known that the linear functions are correlation immune

but not suitable for use in cryptography. There are two measures for nonlinearity of Boolean functions. The algebraic degree is the degree of the algebraic normal form of a Boolean function. Having a high algebraic degree ensures a high linear complexity of the produced key stream and hence better immunity against the Berlekamp Massey shift register synthesis algorithm [9]. A second measure of nonlinearity is the distance from the set of affine functions. A high value of this parameter ensures that the best affine approximation [4] attack will fail. Siegenthaler in [17] proved a fundamental inequality relating the number of variables n, order of correlation immunity m and algebraic degree d of a Boolean function: $m+d \le n$. Moreover, if the function is balanced then $m+d \le n-1$. Also, a balanced mth order correlation immune function is said to be m-resilient. Since it is natural to use balanced functions in stream cipher systems we concentrate only on resilient functions. A resilient Boolean function is said to be optimized if m+d=n-1. The maximum possible nonlinearity (distance from the set of linear functions) for this class of functions is not known. Here we provide construction methods for optimized functions having high nonlinearities. The functions are built using a small set of recursive operations and hence functions on large number of variables are easy to implement using nominal hardware.

Construction procedures for correlation immune (CI) functions were first described by Siegenthaler in [17]. The methods described in [17] are recursive, where a function of (n+1) variables is built from two functions of n variables. Siegenthaler considered two different kinds of constructions, one where the order of correlation immunity remains constant and the other where the order of correlation immunity increases by one at each step. An important spectral characterization of correlation immunity, based on Walsh transform of a Boolean function, was given in [6].

Further attempts at construction was made by Camion et al. in [1], where construction procedure for a certain subset of correlation immune functions were described. In [2], the construction procedure for bent functions is modified to get correlation immune functions. Seberry et al. [16], also provided a method of constructing the same subset as in [1] of correlation immune functions. They also separately considered the algebraic degree, nonlinearity and propagation characteristics of their construction method. The functions constructed in [16] has good nonlinearity for non optimized functions. However, for optimized functions the nonlinearity of [16] decreases. We interpret the direct construction method proposed in [16] in a simpler manner (see Section 5) as a concatenation of linear functions. This interpretation simplifies the proofs related to correlation immunity and nonlinearity.

Evolutionary techniques are applied in [11] to design first order correlation immune balanced functions with high nonlinearity. The technique considers the output column of the function as a string and applies genetic algorithm to manipulate this string. Therefore this technique is difficult to apply to construct functions on n variables for even moderate values of n. Moreover, it is not clear whether these functions optimize the Siegenthaler's inequality. To be precise, by relaxing the optimization criterion of the Siegenthaler's inequality, we can

achieve better nonlinearity than in [11]. Favorable results can also be found using construction procedure in [16].

In another approach to the problem, Filiol and Fontaine [5, Section 5] describe a method to construct functions which achieve a good trade-off between nonlinearity, balancedness, degree and correlation immunity. They identify a 7 variable function f with nonlinearity 56 and degree 6. Using f, in [5, Section 5], they construct a balanced 9 variable function g with nonlinearity 224, correlation immunity of order 2 and degree 6, where they use a technique which was first introduced in [17, Section VI], and later in [1, Corollary 4.1]. The function g is optimized with respect to Siegenthaler's inequality. The function g is so far the best known optimized function on 9 variables with correlation immunity of order 2. The key of this construction is the existence of f.

We use concatenation techniques and introduce generic construction functions (see Definition 5), which recursively build a correlation immune function of (n+1) variables from two correlation immune functions of n variables. We initially start with bent functions which are modified a little to get optimized algebraic degree. A sequence of such constructors is applied to build correlation immune functions of desired orders from non correlation immune balanced Boolean functions with high nonlinearity. The degree of the resulting function is same as that of the initial function. The method can easily be extended to design functions with moderate to large number of input variables using a special representation of the constructed Boolean functions (see Definition 6). The actual trade-off between nonlinearity and correlation immunity is explicit (see Theorem 11). Also Theorem 11 provides a lower bound on the nonlinearity of a function optimized with respect to Siegenthaler's inequality [17].

Both our technique as well as the technique of [16] can be used to construct highly nonlinear, balanced, n variable, mth order correlation immune (m resilient) functions having algebraic degree n-m-1. We show that for all m such that $m+2\log_2(m+3)+3 < n$, the nonlinearity obtained by our method for **optimized functions** is better than that of [16]. Thus as n increases, for almost all m, we obtain a better nonlinearity. Conversely, if we fix an m, then there exists an N, such that for all $n \ge N$, the nonlinearity obtained by our method is better. As examples, using our techniques one can construct

- $1.\ 10$ variable balanced functions with degree 8, order of correlation immunity 1 and nonlinearity 476 and
- 2. 50 variable balanced functions with degree 20, order of correlation immunity 29 and nonlinearity $2^{49} 2^{39} 2^{30}$.

None of the currently known methods can be used to construct such optimized functions. Moreover, there are widely different functions in the constructed class (see Example 1 in Section 6).

Next we provide a list of notations.

1. For strings S_1, S_2 of same length λ , we denote by $\#(S_1 = S_2)$ (respectively $\#(S_1 \neq S_2)$), the number of places where S_1 and S_2 are equal (respectively unequal). The *Hamming distance* between S_1, S_2 is denoted as $D(S_1, S_2)$, i.e. $D(S_1, S_2) = \#(S_1 \neq S_2)$. The Walsh Distance is defined as, $wd(S_1, S_2) = \#(S_1, S_2)$

 $\#(S_1 = S_2) - \#(S_1 \neq S_2)$. Note that, $wd(S_1, S_2) = \lambda - 2D(S_1, S_2)$. Also the *Hamming weight* or simply the weight (number of 1s in S) of S is denoted as wt(S).

2. By Ω_n , we denote the set of all Boolean functions on n variables, i.e., the set of all binary strings of length 2^n . If $f,g\in\Omega_{n-1}$, then F=fg is a function in Ω_n whose output column is the concatenation of the output columns of f and g. Given the truth table of a function f of n input variables $\{X_1,X_2,\ldots,X_n\}$, we also interpret f as a binary string of length 2^n , the output column of the truth table. The first half of the string f is denoted as f^u and the second half is denoted as f^l . If $f\in\Omega_n$, then $f^u,f^l\in\Omega_{n-1}$ and are given by $f^u(X_{n-1},\ldots,X_1)=f(0,X_{n-1},\ldots,X_1)$ and $f^l(X_{n-1},\ldots,X_1)=f(1,X_{n-1},\ldots,X_1)$. In the truth table the column corresponding to an input variable X_j occurs to the left of the column corresponding to the input variable X_i , if j>i. Note that a function $f\in\Omega_n$ may be a non degenerate function of i variables for i< n.

3. The reverse of the string S is denoted by S^r . The bitwise complement of a string S is denoted as S^c . If f is a Boolean function, then f^r , the function obtained by reversing the output column of the truth table is given by $f^r(X_n, \ldots, X_1) = f(1 \oplus X_n, \ldots, 1 \oplus X_1)$, where \oplus denotes the XOR operation. Similarly, the function f^c obtained by complementing each bit of the output column of f is given by $f^c(X_n, \ldots, X_1) = 1 \oplus f(X_n, \ldots, X_1)$.

We next define the important cryptographic properties of Boolean functions for stream cipher applications. These also appear in [5,16,12,17].

Definition 1. A Boolean function f of n variables is said to be linear/affine if f can be expressed as $f = \bigoplus_{i=1}^{n} a_i X_i \oplus b$, where $a_i, b \in \{0, 1\}$ for all i. The set of

linear/affine functions of n variables is denoted as L(n). A Boolean function f of n variables is said to be nonlinear if f is not linear/affine. We denote the measure of nonlinearity of an n variable function f as $nl(f) = min_{g \in L(n)}(D(f,g))$.

Note that $L(n)=\{H\mid H=hh \text{ or } hh^c, h\in L(n-1)\}$. Let $h\in L(n)$ be a non degenerate function of m $(1\leq m\leq n)$ variables. If m is even then $h^r=h$ else if m is odd, $h^r=h^c$. The linear function $h\in L(n)$ is degenerate if m< n. A high nonlinearity ensures that the best affine approximation cryptanalytic attack will fail. (See [4] for a description of this method). It is known [13] that for even n, the maximum nonlinearity achievable by a Boolean function is $nl(f)=2^{n-1}-2^{\frac{n}{2}-1}$. Such functions are called bent functions and their combinatorial properties have been studied [3,4,13]. A simple construction method for bent

functions from [13] is
$$h(X_1, ..., X_p, Y_1, ..., Y_p) = \bigoplus_{i=1}^p X_i Y_i \oplus g(Y_1, ..., Y_p)$$
 where $g \in \Omega_n$ is arbitrary. For odd n , the corresponding class of functions have not

 $g \in \Omega_p$ is arbitrary. For odd n, the corresponding class of functions have not been characterized. (See [15] for some best known examples). Moreover, bent functions are known to be unbalanced and are not correlation immune. Meier and Staffelbach [10] have described a procedure to construct balanced nonlinear functions from bent functions. So if one is looking for functions which optimize Siegenthaler's inequality, one cannot hope to attain the maximum value of nl(f).

Another important criterion is algebraic degree, since it determines the linear complexity of the output sequence of the function (see [4]). The relationship of algebraic degree to the order of correlation immunity was studied in [17,6].

Definition 2. The algebraic degree or simply the degree of $f \in \Omega_n$, denoted by deg(f), is defined to be the degree of the algebraic normal form of f. See [17] for definition of algebraic normal form and its degree.

Siegenthaler [17] was the first to define correlation immune functions from information theoretic point of view using the concept of mutual information.

A function $f(X_1, X_2, ..., X_n)$ is mth order correlation immune [17] if the mutual information $I(X_{i_1}, X_{i_2}, ..., X_{i_m}; Z) = 0$ for all possible choices of m distinct variables $X_{i_1}, X_{i_2}, ..., X_{i_m} \in \{X_1, X_2, ..., X_n\}$, with $1 \leq m \leq n-1$. From [7], this is equivalent to $Prob(Z = 1 \mid X_{i_1} = C_{i_1}, X_{i_2} = C_{i_2}, ..., X_{i_m} = C_{i_m}) = Prob(Z = 1)$ for each of the combinations $C_{i_1}, C_{i_2}, ..., C_{i_m} \in \{0, 1\}$.

A characterization of correlation immunity based on Walsh transform of Boolean functions was obtained in [6]. We first provide the definition of Walsh transform.

Definition 3. Let $\overline{X} = (X_1, ..., X_n)$ and $\overline{\omega} = (\omega_1, ..., \omega_n)$ be n-tuples on GF(2) and $\overline{X}.\overline{\omega} = X_1\omega_1 \oplus ... \oplus X_n\omega_n$. Let $f(\overline{X})$ be a Boolean function whose domain is the vector space over $GF(2)^n$. Then the Walsh transform of $f(\overline{X})$ is a real valued function over $GF(2)^n$ that can be defined as $F(\overline{\omega}) = \sum_{\overline{X}} (-1)^{f(\overline{X}) \oplus \overline{X}.\overline{\omega}}$,

where the sum is over all \overline{X} in $GF(2)^n$.

The following result provides the relationship between Walsh distance and Walsh transform.

Proposition 1.
$$F(\overline{\omega}) = wd(f, \bigoplus_{i=1}^{i=n} \omega_i X_i).$$

The following characterization of correlation immunity, based on Walsh transform, was given in [6].

Theorem 1. ([6]) A function $f(X_n, X_{n-1}, ..., X_1)$ is mth order correlation immune iff its Walsh transform F satisfies $F(\overline{\omega}) = 0$, for $1 \le wt(\overline{\omega}) \le m$.

Proposition 2. Let
$$h, f \in \Omega_n$$
. Then (a) $wd(h, f) = -wd(h^c, f)$ and (b) $wd(h, f^r) = wd(h^r, f)$. Consequently, $wd(h, f) = 0$ iff $wd(h, f^c) = 0$.

We use the following definition of correlation immunity which follows from Proposition 1, Theorem 1 and Proposition 2.

Definition 4. A function $f(X_n, X_{n-1}, ..., X_1)$ is said to be $mth \ (1 \le m \le n-1)$ order correlation immune if wd(f,h) = 0 where $h \in L(n)$ and h is a non degenerate function of i variables with $1 \le i \le m$. Moreover, if f is balanced then f is called m-resilient.

From this definition it is clear that if a function is mth order correlation immune, then it is kth order correlation immune for $1 \le k \le m$. We define,

- 1. $C_n(m) = \{ f \in \Omega_n \mid f \text{ is correlation immune of order } m \text{ but not correlation immune of order } m+1 \}.$
- 2. $A_n(m) = \bigcup_{m \le k \le n-1} C_n(k)$, is the set of all correlation immune functions of order m or more.
- 3. A function is called *correlation immune* if it is at least correlation immune of order one. Also $A_n = A_n(1)$, is the set of all correlation immune functions of n variables.

Let f be a balanced function of degree d, and $f \in C_n(m)$. Then f is optimized with respect to balancedness, degree and order of correlation immunity if m + d = n - 1. The maximum value of nl(f) for such functions is not known. In Theorem 11 we describe methods to construct such optimized functions with sufficiently large values of nl(f). We next define three constructions P, Q, R as follows. These constructions have also been used in [8] to obtain the currently best known lower bounds on (balanced) correlation immune Boolean functions.

Definition 5. 1.
$$P: \Omega_{n-1} \times \Omega_{n-1} \to \Omega_n, \ P(f,g) = f^u g^u g^l f^l$$
.
2. $Q: \Omega_{n-1} \times \Omega_{n-1} \to \Omega_n, \ Q(f,g) = fg = f^u f^l g^u g^l$.
3. $R: \Omega_{n-1} \times \Omega_{n-1} \to \Omega_n, \ R(f,g) = f^u g^u f^l g^l$.

Later we will use these constructions to recursively build correlation immune functions. The construction Q appears in [17], although in a different form. Note that, the generic construction functions P, Q, R should not be viewed as linear combination of two Boolean functions. As example, if we consider the Boolean function $Q(f, f^r)$, then the nonlinearity of $Q(f, f^r)$ will be twice that of f and the number of terms with highest algebraic degree will increase. We discuss it elaborately in the next section.

2 Nonlinearity, Algebraic Degree, and Balancedness

We provide a few technical results in this section related to nonlinearity, algebraic degree and balancedness.

Theorem 2. Let
$$f, g \in \Omega_{n-1}$$
 and $F = \Psi(f, g)$ where $\Psi \in \{P, Q, R\}$. Then $nl(F) \ge nl(f) + nl(g)$. Moreover, if $g = f$, $g = f^c$ or $g = f^r$, then $nl(F) = nl(f) + nl(g) = 2nl(f)$.

Next we state without proof the following result on the degree of the constructed function. The proof consists in checking the different cases.

Theorem 3. Let $f \in \Omega_n$ and $F = \Psi(f, f^{\tau})$, where $\Psi \in \{P, Q, R\}$ and $\tau \in \{c, r\}$. Then, deg(F) = deg(f).

The special case of Theorem 3 with $\Psi=Q$ and $\tau=c$ was mentioned in [5]. The importance of this result lies in the fact that the degree of the constructed function is equal to the degree of the original function. It is known [13] that the degree of bent functions of n variables for $n\geq 4$ is at most $\frac{n}{2}$. We propose the following simple but *powerful* method to improve the degree. Note that, $X_1 \ldots X_n$ means logical AND of X_1 to X_n .

Theorem 4. Let $h \in \Omega_n$ be of degree less than n and $f = h \oplus X_1 \dots X_n$. Then deg(f) = n and $nl(f) \ge nl(h) - 1$. Moreover, if h is a bent function then nl(f) = nl(h) - 1.

Proof. Let, $g \in L(n)$. Then D(f,g) is either D(h,g)-1 or D(h,g)+1. If h is bent, $nl(f) \leq nl(h)$, and so nl(f) = nl(h)-1.

If we start with a bent function $h \in \Omega_8$ and use the above theorem then we can get a function $f \in \Omega_8$ of degree 8 and nonlinearity 119. Using this f, one can get $F = \Psi(f, f^c) \in \Omega_9$, which is balanced, has degree 8 and nonlinearity 238. Generalizing, we can get balanced functions $F \in \Omega_{2p+1}$ having degree 2p and nonlinearity $2^{2p} - 2^p - 2$. We now discuss the following negative results which can be obtained from Siegenthaler's inequality. Let $f \in \Omega_n$.

- 1. If deg(f) = n 1, then f is not both correlation immune and balanced.
- 2. If deg(f) = n, then f is neither correlation immune nor balanced.

Both 1 and 2 follow from Siegenthaler's inequality $m + d \le n - 1$ for balanced functions $f \in C_n(m)$ having degree d. To see that if deg(f) = n, then f is not balanced, suppose the converse, i.e., deg(f) = n and f is balanced. Since f is balanced, using Theorem 8 in the next section, $Q(f, f^c) = ff^c \in C_{n+1}(1)$. Also, ff^c is balanced and has degree n. Thus, Siegenthaler's inequality is violated for $ff^c \in \Omega_{n+1}$.

Note that item 2 shows that a function of n variables cannot both have degree n and be balanced. Thus it relates two simple properties of Boolean functions. However, it requires the use of correlation immunity, which is a much more specialized property. This shows that there cannot exist balanced functions $F \in \Omega_n$ of degree n. Filiol and Fontaine [5] provided examples of balanced $F \in \Omega_9$ having nonlinearity 240 but degree upto 7. It is interesting to find out whether there exists balanced $F \in \Omega_9$, having degree 8 and nonlinearity 240.

Theorem 4 shows that the degree can be increased significantly with insignificant change in nonlinearity. Moreover, it can be checked that though f in the above theorem has only one term of degree n, the number of terms of degree n in $\Psi(f, f^r) \in \Omega_{n+1}$ is more than one. We state one specific result regarding this.

Proposition 3. Let $f \in \Omega_n$ with degree n. Then $Q(f, f^r) \in \Omega_{n+1}$ contains n terms of degree n.

The linear complexity of the output sequence produced by the Boolean function depends on the algebraic normal form of the function and the lengths of the input LFSRs [14,4]. Having more terms of degree n ensures that the linear complexity of the output sequence is higher. See Example 1 in the last section

for further illustration regarding the number of high degree terms. Proper use of this technique will ensure that the functions designed using Construction 1 (see later), will have this property. This has direct implication towards the stability of the generated sequence [4]. We would like to point out that this phenomenon does not hold for the construction $Q(f, f^c)$ given in [17,5]. Next, we list a few simple results on balancedness.

Proposition 4. (a) A function of the form ff^c is balanced. (b) If f is a balanced function then both f^r and f^c are balanced. (c) Let $f, g \in \Omega_n$ be two balanced functions, and $F = \Psi(f, g)$, where $\Psi \in \{P, Q, R\}$. Then F is also balanced.

3 Correlation Immunity

Here we provide generalized construction methods for correlation immune functions. First we state the following two results which have been proved in different forms in [12,1] and [17] respectively.

Proposition 5. Let $h \in \Omega_n$. Then $Q(h, h^r) = hh^r \in A_{n+1}$.

Proposition 6. Let $f \in \Omega_n$. Then $Q(f, f^c) = ff^c \in A_{n+1}$ iff f is balanced.

Next we state without proof the following basic result.

Lemma 1. Let $f \in A_n(m)$ (respectively $C_n(m)$). Then $f^r, f^c \in A_n(m)$ (respectively $C_n(m)$).

In [17, Section IV] Siegenthaler proposed a construction of $F \in A_{n+1}(m)$ from $f, g \in A_n(m)$ as follows.

Theorem 5. ([17]) If $Z_1 = f_1(X_1, X_2, ..., X_n)$ and $Z_2 = f_2(X_1, X_2, ..., X_n)$ are mth-order correlation immune functions of n binary variables such that $Prob(Z_1 = 1) = Prob(Z_2 = 1)$, then the binary-valued function f of n + 1 random variables defined by the GF(2) expression $f(X_1, X_2, ..., X_{n+1}) = X_{n+1} f_1(X_1, X_2, ..., X_n) + (X_{n+1} + 1) f_2(X_1, X_2, ..., X_n)$

 $f(X_1, X_2, ..., X_{n+1}) = X_{n+1} f_1(X_1, X_2, ..., X_n) + (X_{n+1} + 1) f_2(X_1, X_2, ..., X_n)$ is also mth order correlation immune.

The condition $Prob(Z_1 = 1) = Prob(Z_2 = 1)$ is equivalent to the condition $wt(f_1) = wt(f_2)$. Note that the construction in the above theorem corresponds to our construction Q. We further generalize the construction to include P, R also.

Lemma 2. Let $f, g \in A_n(m)$ and F be of the form $F = P(f, g) = f^u g^u g^l f^l$. If (a) m = 1 or (b) m > 1 and wt(f) = wt(g), then $F \in A_{n+1}(m)$.

Proof. Let f, g be functions of $\{X_1, X_2, \ldots, X_n\}$ and F be a function of $\{X_1, X_2, \ldots, X_{n+1}\}$. We use the characterization of correlation immunity given in Definition 4. Let us consider any linear/affine function $H \in L(n+1)$, where H is a non degenerate function of k variables $(1 \le k \le m)$.

Now we will have four cases.

- 1. If H contains k variables from $\{X_1, X_2, \ldots, X_{n-1}\}$ then H is of the form hhhh. Now, $wd(F, H) = wd(f^ug^ug^lf^l, hhhh) = wd(f, hh) + wd(g, hh) = 0$, as f, g are mth order correlation immune.
- 2. If H contains X_n and the remaining k-1 variables from $\{X_1, X_2, \ldots, X_{n-1}\}$ then H is of the form hh^chh^c . Then, $wd(F, H) = wd(f^ug^ug^lf^l, hh^chh^c) = wd(f, hh^c) + wd(g, h^ch) = 0$.
- 3. If H contains X_{n+1} and the remaining k-1 variables from $\{X_1, X_2, \ldots, X_{n-1}\}$ then H is of the form hhh^ch^c . Now, $wd(F, H) = wd(f^ug^ug^lf^l, hhh^ch^c) = wd(f, hh^c) + wd(g, hh^c) = 0$.
- 4. If H contains X_n, X_{n+1} and the remaining k-2 variables from $\{X_1, X_2, \ldots, X_{n-1}\}$ then H is of the form hh^ch^ch . Now two cases arise. (a) If k-2>0, then $wd(F,H)=wd(f^ug^ug^lf^l,hh^ch^ch)=wd(f,hh)+wd(g,h^ch^c)=0$.
 - (b) If k-2=0, then H is of the form $0^{n-1}1^{n-1}1^{n-1}0^{n-1}$ and hence, $wd(F,H)=wd(f^ug^ug^lf^l,0^{n-1}1^{n-1}1^{n-1}0^{n-1})=wd(f,0^n)+wd(g,1^n)=0$, if wt(f)=wt(g). Note that the weight condition is not required if m=1.

Hence by Definition 4, F is mth order correlation immune.

The case for the construction R is similar. Hence we get,

Theorem 6. Let $f, g \in A_n(m)$, with wt(f) = wt(g) and $F = \Psi(f, g)$, where $\Psi \in \{P, Q, R\}$. Then $F \in A_{n+1}(m)$.

In [17] only a construction with two correlation immune functions f, g of same order was considered. However, if the correlation immunity of f, g are of different orders then we get the following result.

Theorem 7. Let $f \in C_n(m_1)$ and $g \in A_n(m_2)$ with $m_1 < m_2$. Then $F \in C_{n+1}(m_1)$ if (a) $\Psi = P$ and $m_1 = 1$ or (b) $\Psi = P$, Q or R and wt(f) = wt(g).

Proof. The proof that F belongs to $A_{n+1}(m_1)$ is similar to the above theorem. It can be checked that if $m_1 = 1$ then the weight condition wt(f) = wt(g) is not required for P. To see that $F \in C_{n+1}(m_1)$, note that there exists a function $h \in L(n)$, which is non degenerate of $(m_1 + 1)$ variables such that $wd(f,h) \neq 0$ but wd(g,h) = 0. Depending on Ψ we can use this h to build a linear function $H \in L(n+1)$ which is non degenerate of $(m_1 + 1)$ variables such that $wd(F,H) \neq 0$. Hence F is not correlation immune of order $(m_1 + 1)$. \square

Next we consider construction of (m + 1)th order correlation immune function from mth order correlation immune functions.

Proposition 7. Let f be an n variable balanced function with mth order correlation immunity. Then $F = Q(f, f^c) = f f^c$ is an (n+1) variable function with (m+1)th order correlation immunity.

In a different form, this was first observed in [17] and later in [1]. This is the basic technique of construction used in [5]. We show that the same result can be achieved using R also.

Theorem 8. Let $f \in C_n(m)$ and $F = \Psi(f, f^c)$ where $\Psi \in \{Q, R\}$. Then $F \in C_{n+1}(m+1)$ iff f is balanced. Moreover, F is balanced.

Proof. We prove this theorem for $\Psi=R$, the other case being similar. Let us consider any linear/affine function $H\in L(n+1)$ which is a non degenerate function of k variables $(1\leq k\leq m+1)$. For $(1\leq k\leq m)$ the proof that wd(F,H)=0 is similar to that of Lemma 2. If H is a non degenerate function of (m+1) variables then H can be of the forms $hhhh, hhh^ch^c, hh^chh^c$ and hh^ch^ch . Let H be of the form hh^chh^c , where $h\in L(n-1)$ is non degenerate of m variables. So, $wd(R(f,f^c),H)=wd(R(f,f^c),hh^chh^c)=wd(f,hh)+wd(f^c,h^ch^c)=2wd(f,hh)=0$ as $f\in C_n(m)$ and hh is a non degenerate function of m variables. It can be checked that for the other cases also $wd(R(f,f^c),H)=0$. This shows that $F\in A_{n+1}(m+1)$.

The resulting $R(f, f^c)$ will not be in $A_{n+1}(m+2)$. We show a function $H \in L(n+1)$ which is a non degenerate function of (m+2) variables, such that $wd(R(f, f^c), H) \neq 0$. Since f is not correlation immune of order (m+1), there exists a non degenerate function $h_1 \in L(n)$ of (m+1) variables such that $wd(f, h_1) \neq 0$. Now two cases arise.

Case 1: h_1 is of the form hh, where $h \in L(n-1)$. Then h is nondegenerate of (m+1) variables and let $H \in L(n+1)$ be of the form hh^chh^c . Then, $wd(R(f, f^c), H) = wd(f^u(f^u)^c f^l(f^l)^c, hh^chh^c) = wd(f, hh) + wd(f^c, h^ch^c) = 2wd(f, hh) \neq 0$.

Case 2: h_1 is of the form hh^c , where $h \in L(n-1)$. In this case h is non degenerate of m variables and take $H \in L(n+1)$ to be of the form hh^ch^ch . Now, $wd(R(f, f^c), H) = wd(f^u(f^u)^cf^l(f^l)^c, hh^ch^ch) = wd(f, hh^c) + wd(f^c, h^ch) = 2wd(f, hh^c) \neq 0$.

The above result does not in general hold for the construction P. If h_1 in the above proof is of the form hh^c , and we choose H to be of the form hh^chh^c , which is non degenerate of (m+1) variables, then

 $wd(P(f, f^c), H) = wd(f^u(f^u)^c(f^l)^c f^l, hh^c hh^c) = wd(f, hh^c) + wd(f^c, h^c h) = 2wd(f, hh^c) \neq 0$. However, the following result holds.

Lemma 3. Let $f \in \Omega_n - A_n$ be such that $wt(f^u) = wt(f^l)$. Then $P(f, f^c) \in A_{n+1}$ and is balanced.

If f is a correlation immune function of even order then we can use f^r instead of f^c in Theorem 8.

Theorem 9. Let $f \in C_n(m)$ and $\Psi \in \{Q, R\}$.

1. Let $F = \Psi(f, f^r)$. Then, $F \in C_{n+1}(m+1)$ iff m is even. Moreover, F is balanced iff f is balanced.

2. Let $F = \Psi(f, (f^r)^c)$. Then, $F \in C_{n+1}(m+1)$ iff m is odd. Moreover, F is balanced.

Proof. We only prove (1) for $\Psi = R$. We show that if $H \in L(n+1)$, and H is a non degenerate function of k $(1 \le k \le m+1)$ variables, then $wd(R(f, f^r), H) = 0$. The case where $1 \le k \le m$ is similar to Lemma 2. Now for k = m+1 four cases arise.

- 1. H is of the form hhhh. Then $h \in L(n-1)$ and h is a non degenerate function of (m+1) variables. Since m is even, (m+1) is odd and so $h^r = h^c$. Therefore, $wd(R(f, f^r), hhhh)) = wd(f, hh) + wd(f^r, hh) = wd(f, hh) + wd(f, h^rh^r) = wd(f, hh) + wd(f, h^ch^c) = wd(f, hh) wd(f, hh) = 0$.
- 2. H is of the form hh^chh^c . Then hh^c is a non degenerate function of (m+1) variables and hence h is a non degenerate function of m variables. Therefore, $wd(R(f,f^r),hh^chh^c)=wd(f,hh)+wd(f^r,h^ch^c)=0+0=0$ as $f,f^r\in C_n(m)$.
- 3. H is of the form hhh^ch^c . Then hh is a non degenerate function of m variables and hence hh^c is a non degenerate function of (m+1) variables. Therefore, $wd(R(f, f^r), hhh^ch^c) = wd(f, hh^c) + wd(f^r, hh^c) = wd(f, hh^c) + wd(f, (hh^c)^r) = wd(f, hh^c) wd(f, hh^c) = 0$.
- 4. H is of the hh^ch^ch . Then h is a non degenerate function of (m-1) variables and so hh^c is a non degenerate function of m variables. Hence, $wd(R(f, f^r), hh^ch^ch) = wd(f, hh^c) + wd(f^r, h^ch) = 0 + 0 = 0$.

Hence $wd(R(f, f^r), H) = 0$ and so $R(f, f^r) \in A_{n+1}(m+1)$. The proof that $R(f, f^r) \notin A_{n+1}(m+2)$ is similar to Theorem 8. If m is odd, then it can be checked that $F \notin C_{n+1}(m+1)$.

Camion et al. [1] had earlier proved one side of both (1) and (2) of the above theorem for $\Psi = Q$ only.

Remark 1. In Theorem 8 and Theorem 9 we can obtain a weaker result by replacing $C_n(m)$ and $C_{n+1}(m+1)$ by $A_n(m)$ and $A_{n+1}(m+1)$ respectively.

We also have the following result which is similar to Lemma 3.

Lemma 4. Let $f \in \Omega_n - A_n$ be such that $wt(f^u) = wt(f^l)$. Then $P(f, f^r) \in A_{n+1}$.

We will be using the results of Section 2 and Section 3 to design cryptographically strong Boolean functions in the next section.

4 Generalized Construction

Here we describe a recursive procedure to design highly nonlinear Boolean functions which optimizes balancedness, degree and order of correlation immunity. Such functions are ideally suited for stream cipher applications since they can resist all known types of attacks. First we require the following definition. We use the convention that $f^{rc} = (f^c)^r = (f^r)^c$.

Definition 6. Let $(S_i)_{1 \leq i \leq q}$ be a finite sequence, where, $S_i \in \{Q, R\} \times \{c, r, rc\}$. Given a function $h \in \Omega_k$ and a sequence S_i of length q we define a function $F \in \Omega_{q+k}$ as follows.

 $F_0 = h$ and $F_i = \Psi_i(F_{i-1}, F_{i-1}^{\tau_i})$ where $S_i = (\Psi_i, \tau_i)$, for $i \geq 1$, and $F = F_q$. We say that F is represented by (h, S_1, \ldots, S_q) and the length of the representation is q. First we observe that given a function $h \in \Omega_k$ it is easy to design a linear time (on the number of inputs to the function) algorithm that generates a function $F \in \Omega_{q+k}$ represented by (h, S_1, \ldots, S_q) . Though the size of the function F may be large, we need not store the whole truth table for F. Using the representation of F, the storage space required is not much larger than h. The penalty is that we require an algorithm to calculate the output of F. This can be done in O(q) time (specifically, q clocks in hardware circuit) if h is implemented as a truth table. However, very low cost pipelined circuit (using flip flops) can be developed which produces a output at each clock pulse after an initial latency period of q clocks. Both the hardware and the algorithm are interesting which we omit here due to space constraint. Now we state some important properties of functions constructed by the above procedure.

Theorem 10. Let $h \in \Omega_k$ and $F \in \Omega_{m+k+1}$ be represented by $(h, S_1, \ldots, S_{m+1})$ where $S_i = (\Psi_i, \tau_i)$, $\tau_{2i+1} \in \{c, rc\}$ and $\tau_{2i+2} \in \{c, r\}$ for $i \geq 0$. Then F is balanced and (1) $nl(F) = 2^{m+1}nl(h)$ (2) deg(F) = deg(h), (3) If $m \geq 1$, then $F \in A_{m+k+1}(m)$. Moreover, if degree of h is k, then $F \in C_{m+k+1}(m)$.

Proof. (1) Follows from Theorem 2. (2) Follows from Theorem 3. (3) Follows from Theorem 8, Theorem 9 and Remark 1. Moreover, if degree of h is k, then F can not be correlation immune of order m+1 due to the Siegenthaler's inequality.

Note that there are four possible options of S_i for i > 0. Moreover, the construction P can also be used in the first step S_1 , since the purpose of the first step is to attain balancedness. This generalizes the construction method of [5, Section 5], which uses the sequence $S_i = (Q, c)$ for all $i \geq 1$.

Corollary 1. Let $h \in \Omega_k$ be balanced and $F \in \Omega_{m+k}$ be represented by (h, S_1, \ldots, S_m) , where $S_i = (\Psi_i, \tau_i)$, $\tau_{2i+1} \in \{c, r\}$ and $\tau_{2i+2} \in \{c, rc\}$ for $i \geq 0$. Then F is in $A_{m+k}(m)$. Moreover, if degree of h is (k-1), the maximum degree attained for a balanced function, then $F \in C_{m+k}(m)$.

It is important to realize that there are different trade-offs involved among the parameters, algebraic degree deg(.), order of correlation immunity m, nonlinearity nl(.), balancedness and the number of input variables n. The first result from [17], is that for any Boolean function f, $deg(f) + m \le n$ and for balanced Boolean functions, $deg(f) + m \le n - 1$. The next result is that the maximum value of nonlinearity for even n is achieved for bent functions and it is known [13] that for $n \ge 4$, the degree of such functions cannot exceed $\frac{n}{2}$. Let us now consider the following construction which provides a good trade-off among the parameters.

Construction 1. On input n, m we provide a method to construct a balanced n variable mth order correlation immune function with algebraic degree k = n - m - 1. Let $h \in \Omega_k$ of degree k be as follows.

If k is even, then h is formed by adding the term $X_1 ... X_k$ (logical AND of X_1 to X_k) to a bent function g of k variables. If k is odd then h is formed by adding the term $X_1 ... X_k$ to a function g of k variables, where g is formed by

concatenating two bent functions of (k-1) variables.

Let $F \in \Omega_n$ where n = m+k+1 and $m \ge 1$. F is represented by $(h, S_1, \ldots, S_{m+1})$ where $S_i = (\Psi_i, \tau_i)$, $\Psi_i \in \{Q, R\}$, $\tau_{2i+1} \in \{c, rc\}$ and $\tau_{2i+2} \in \{c, r\}$ for $i \ge 0$.

It is clear from the above discussion that Construction 1 provides functions which optimize Siegenthaler's inequality. We now find out the exact expression of nonlinearity obtained by the above construction. The result follows from Theorem 10, Corollary 1 and the nonlinearity of bent functions.

Theorem 11. Consider $F \in C_n(m)$ as in Construction 1.

- (1) If $n \not\equiv m \mod 2$, then $nl(F) = 2^{n-1} 2^{\frac{n+m-1}{2}} 2^{m+1}$.
- (2) If $n \equiv m \mod 2$ then $nl(F) \geq 2^{n-1} 2^{\frac{n+m}{2}} 2^{m+1}$.

Proof. (1) We take a bent function g of k=n-m-1 variable and $h=g\oplus X_1X_2\dots X_{n-m-1}$. Thus by Theorem 4, $nl(g)=2^{n-m-2}-2^{\frac{n-m-1}{2}-1}-1$. Then we apply our method of Definition 6 to get $nl(f)=2^{m+1}$ $(2^{n-m-2}-2^{\frac{n-m-1}{2}-1}-1)$. (2) We take a bent function g_1 of n-m-2 variables. Then we use bent concatenation to get g of n-m-1 variables with nonlinearity $nl(g)=2^{n-m-2}-2^{\frac{n-m-2}{2}}$. Now, $h=g\oplus X_1X_2\dots X_{n-m-1}$. Thus, $nl(h)\geq 2^{n-m-2}-2^{\frac{n-m-2}{2}}-1$. Hence, $nl(f)\geq 2^{m+1}$ $(2^{n-m-2}-2^{\frac{n-m-2}{2}}-1)$. □

This also shows that by varying the order of correlation immunity, one can adjust the nonlinearity of the optimized functions.

5 Direct Construction

Here we provide a simpler interpretation of the construction method provided in [16] and show that this also gives simpler proofs for the order of correlation immunity and nonlinearity of the constructed functions. Let L(n,k) be the set of all $f \in L(n)$, which are the sum modulo 2 (XOR) of exactly k variables and $MU(n,k) = L(n,k) \cup L(n,k+1) \cup \ldots \cup L(n,n)$. Also let $ML(n,k) = L(n,1) \cup L(n,2) \cup \ldots \cup L(n,k)$.

Definition 7. Let $n = n_1 + n_2$ and choose 2^{n_1} functions $f_0, \ldots, f_{2^{n_1}-1}$ from the set $MU(n_2, m+1)$. Let $f = f_0 \ldots f_{2^{n_1}-1}$, and denote by $\Gamma(n, n_2, m)$ the set of all such functions. Clearly $\Gamma(n, n_2, m) \subseteq \Omega_n$.

We first state a simple result which is crucial to understand the cryptographic properties of the construction provided by Definition 7. The proof is a simple consequence of the fact that the XOR of two linear functions is also a linear function.

Proposition 8. Let $l_1, l_2 \in L(n)$. (a) If $l_1 = l_2$ then $D(l_1, l_2) = 0$. (b) If $l_1 = l_2^c$ then $D(l_1, l_2) = 2^n$ and (c) If $l_1 \neq l_2$ or l_2^c then $D(l_1, l_2) = 2^{n-1}$. Consequently, the Walsh distances are respectively, $2^n, -2^n$ and 0.

The following result is easy to see.

Proposition 9. The construction provided in Definition 7 is same as that given by Equation 5 of [16].

This proves (using [16, Corollary 8]) that any function in $\Gamma(n, n_2, m)$ is an mth order CI function. Here we provide a much simpler direct proof as follows.

Theorem 12. $\Gamma(n, n_2, m) \subseteq A_n(m)$.

Proof. Let $f \in \Gamma(n, n_2, m)$. We show that for any $l \in L(n, k)$, wd(f, l) = 0 for all $1 \le k \le m$. We write $f = f_0 \dots f_{2^{n_1}-1}$, where each $f_i \in MU(n_2, m+1)$. It is not difficult to see that l can be written as $l_0 \dots l_{2^{n_1}-1}$, where each $l_i \in ML(n_2, m)$. Now $wd(f, l) = wd(f_0 \dots f_{2^{n_1}-1}, l_0 \dots l_{2^{n_1}-1}) = \sum_{i=0}^{2^{n_1}-1} wd(f_i, l_i) = 0$, using Proposition 8, since $f_i, l_i \in L(n_2)$ and $f_i \ne l_i$ or l_i^c .

Visualizing the construction as above, it is easy to obtain the nonlinearity as follows.

Theorem 13. Let $f \in \Gamma(n, n_2, m)$ be of the form $f_0 \dots f_{2^{n_1}-1}$, where each $f_i \in MU(n_2, m+1)$. Then $nl(f) \geq 2^{n-1} - t2^{n_2-1}$, where t is the maximum number of times a function h or its complement h^c are together repeated in the construction $f_0 \dots f_{2^{n_1}-1}$ for some $h \in MU(n_2, m+1)$.

Proof. Let $l \in L(n)$. We have to show that D(f, l) is at least as large as the given bound. Note that l can be written as $l_0 \dots l_{2^{n_1}-1}$, where each l_i is either g or g^c for some $g \in L(n_2)$. Then at most t of the l_i 's and f_i 's can be equal. Using Proposition 8, it follows $D(l, f) \geq (2^{n_1} - t)2^{n_2-1} = 2^{n-1} - t2^{n_2-1}$.

Theorem 13 is first proved in [16, Theorem 14]. However, our proof is much shorter and clearer. One can show as in [16, Theorem 12], that the degree of such functions is $n - n_2 + 1$, provided there are at least two functions g_1, g_2 among the f_i 's of Theorem 13, such that $g_1 \neq g_2$ or g_2^c and there is a variable which occurs in an odd number of these f_i 's. Thus maximum degree is attained if $n_2 = m + 2$, in which case the constructed function optimizes Siegenthaler's inequality.

Let us now estimate the nonlinearity of functions constructed using the method of [16], for functions which optimize Siegenthaler's inequality.

Let $\Omega_{k,n} = MU(n, k+1)$. By nld(n), we denote the lower bound on nonlinearity of n-variable optimized functions achieved by the direct construction of Definition 7 (see also [16]). Note that Siegenthaler's inequality is optimized if $n_2 = m + 2$ and in this case,

 $|\Omega_{m,m+2}| = {m+2 \choose m+1} + {m+2 \choose m+2} = m+3$. Since one has to choose 2^{n_1} functions

from $\Omega_{m,m+2}$, the repetition factor t is at least $\lceil \frac{2^{n-m-2}}{m+3} \rceil$ and hence the nonlinearity obtained is

$$nld(n) = 2^{n-1} - \lceil \frac{2^{n-m-2}}{m+3} \rceil \ 2^{m+1}.$$

Remark 2. The construction method provided in Section 4 is a recursive concatenation of highly nonlinear Boolean functions. On the other hand, the construction provided in Definition 7 is a direct concatenation of linear functions.

6 Results and Comparison to Previous Research

First we compare nld(n), the nonlinearity of [16], with our method. For n variable functions, let the lower bound of nonlinearity obtained by our recursive construction be nlr(n).

1. When
$$n \not\equiv m \mod 2$$
. $nlr(n) = 2^{n-1} - (2^{\frac{n-m-3}{2}} + 1) \ 2^{m+1}$. $nld(n) = 2^{n-1} - \lceil \frac{2^{n-m-2}}{m+3} \rceil \ 2^{m+1}$. So, our method works favorably when

$$\left\lceil \frac{2^{n-m-2}}{m+3} \right\rceil > 2^{\frac{n-m-3}{2}} + 1.$$
 (I)

We consider it more conservatively, i.e., we replace $(2^{\frac{n-m-3}{2}}+1)$ by $2^{\frac{n-m-2}{2}}$. Hence, our method performs better when,

$$\left\lceil \frac{2^{n-m-2}}{m+3} \right\rceil > 2^{\frac{n-m-2}{2}}, \text{ i.e., } \left\lceil \frac{2^{n-m-2}}{2^{\log_2(m+3)}} \right\rceil > 2^{\frac{n-m-2}{2}}, \text{ i.e., } n-m-2 - \log_2(m+3) > \frac{n-m-2}{2}, \text{ i.e., when,} \\
n > m + 2\log_2(m+3) + 2. \tag{IA}$$

2. When
$$n \equiv m \mod 2$$
.
 $nlr(n) = 2^{n-1} - (2^{\frac{n-m-2}{2}} + 1) \ 2^{m+1}$. $nld(n) = 2^{n-1} - \lceil \frac{2^{n-m-2}}{m+3} \rceil \ 2^{m+1}$. So, our method works favorably when $\lceil \frac{2^{n-m-2}}{m+3} \rceil > 2^{\frac{n-m-2}{2}} + 1$. (II)

We consider it more conservatively, i.e., we replace $(2^{\frac{n-m-2}{2}}+1)$ by $2^{\frac{n-m-1}{2}}$. Hence, our method performs better when

One can look at (I) and (II) in two ways.

- 1. If we fix a particular value of m, then there is a certain N, such that nlr(n) > 1nld(n) for all n > N. For example for m = 1, our method performs better for all $n \geq 8$.
- 2. Similarly, if we fix a value of n, we get an upper bound M(n) on m, such that for all $m \leq M(n)$, we have nlr(n) > nld(n). Moreover, from (IA) and (IIA), it is clear that this upper bound M(n) becomes close to n, as n increases.

This clearly shows that in a majority of cases the functions obtained by our method are better than those obtained using [16]. It should also be noted that if we take m = 1, then $nld(n) = 2^{n-1} - 2^{n-3}$. Whereas,

(1) If
$$n$$
 even, then $nlr(n) = 2^{n-1} - 2^{\frac{n}{2}} - 4$. (2) If n odd, then $nlr(n) \ge 2^{n-1} - 2^{\frac{n+1}{2}} - 4$.

It should be noted that high nonlinearity can be obtained by the direct construction provided in [16] without optimizing the Siegenthaler's inequality. Currently no known general method can provide balanced CI functions with such nonlinearity. However, the nonlinearity of this method decreases when the optimization criterion is considered. From [16, Theorem 12, 14], if one does not want to optimize the Siegenthaler's inequality, then the nonlinearity for first order correlation immune functions is (we denote it as nlx(n) for n variable function) $nlx(n) = 2^{n-1} - min_{3 \le r < n} (\lceil \frac{2^{n-r}}{2^r - r - 1} \rceil 2^{r-1})$ with algebraic degree n - r + 1. In the following table we compare nonlinearities of first order CI functions. In second, third and fourth columns we respectively provide nonlinearities of nonoptimized (m+d < n-1) functions constructed using the method of [16], optimized functions constructed using the method of [16] and optimized functions constructed using our recursive method. Each of the entries are <nonlinearity, algebraic degree>. Note that the Equations (IA), (IIA) provides a clear analysis of when our nonlinearity is better than that of [16]. The table only illustrates this point for small values of n.

							nlr(n)
8	112, 5	96, 6	108, 6	11	992, 7	768, 9	956, 9
9	240, 5	192, 7	220, 7	12	1984, 7	1536, 10	1980, 10
10	480, 6	384, 8	476, 8	13	4032, 7	3072, 11	3964, 11

Let us consider the class of all optimized functions constructed using the method of [16] for each n. Then also the maximum (lower bound on) nonlinearity achieved is 96, 208, 448, 896, 1792, 3584 for n from 8 to 13 respectively. Column 4 of the table shows that the nonlinearities obtained by our method are better.

Example 1. The class of functions constructed by our recursive method contains significantly different functions. As a simple example for n=10, and m=1, let $f_1=(h,(Q,c),(Q,r))$ and $f_2=(h,(R,c),(R,r))$, where $h\in\Omega_8$ and is modified from bent functions as in Construction 1. As a concrete example, $h=\bigoplus_{i=1}^4 X_iY_i\oplus X_1X_2X_3X_4\oplus X_1\dots X_4Y_1\dots Y_4$. Then both f_1,f_2 contains 8 terms of degree 8. Moreover, the function $f_1\oplus f_2$ is nondegenerate and contains 14 terms of degree 8. The algebraic normal forms of f_1,f_2 and $f_1\oplus f_2$ are complicated and too long to be written down here.

Next we compare the performance of our construction with [5]. In [5, Section 5], balanced $g \in \Omega_9$ with nl(g) = 224, correlation immunity of order 2 and degree 6 has been reported. The function is optimized with respect to Siegenthaler's inequality. The function g has the representation (f,(Q,c),(Q,c)), where $f \in$ Ω_7 and has degree 6 (only one term) and nonlinearity 56. It was remarked in [5, Example 5] that g is the representative of all such functions obtained which are well-suited for stream cipher application. Using this function f as our initial function, one can construct more functions of the form (f, S_1, S_2) as in Corollary 1, with the same parameters as g above. As an example one can construct a function of the form (f,(Q,r),(Q,c)), which contains 6 terms of degree 6. However, it seems difficult to get such good functions f for a larger number of input variables. The particular function $f \in \Omega_7$ reported in [5, Section 5] was obtained by exhaustive search over a particular subset (the idempotents) of Boolean functions. It seems infeasible to carry out such an exhaustive search for functions of larger number of input variables. Using our method from scratch, if one starts with a bent function $f_1 \in \Omega_6$ and apply Construction 1, we get a balanced, second order correlation immune function g_1 with degree 6 and $nl(g_1) = 216$. The direct construction method in [16] provides a nonlinearity $2^{9-1} - \left\lceil \frac{2^{9-2-2}}{2+3} \right\rceil \ 2^{2+1} = 200.$

We next compare the nonlinearities obtained in [11] with the following simple construction. Algebraic degree and optimization criteria is not considered in [11]

and hence we also do not consider it for this comparison. For n even, we start with a bent function h of (n-2) variables and construct F represented by (h, S_1, S_2) as in Theorem 10. Then F is a balanced correlation immune function of order 1 and nl(F) = 4nl(h). For n odd, we start with a best known example of balanced nonlinear function h of (n-1) variables as in [11, Table 1] and construct F represented by any sequence of length one. Then by Corollary 1, F is a balanced first order correlation immune function with nl(F) = 2nl(h). We compare the result using 3 tuples (n, Nonlinearity [11], Our Nonlinearity) : (8, 112, 112), (9, 232, 232), (10, 476, 480), (11, 976, 984), (12, 1972, 1984). Note that, better nonlinearity for functions of 10 variables and onwards can be found using a deterministic technique compared to an evolutionary one.

The recursive method proposed here can be used effectively to construct functions with large number of variables. As an example, if we take a bent function $h \in \Omega_{20}$ and consider a F represented by a sequence (f, S_1, \ldots, S_{30}) satisfying Theorem 11, then $F \in A_{50}(29)$ with nonlinearity $2^{49} - 2^{39} - 2^{30}$ and deg(F) = 20. Currently, there are no known methods which can construct such an optimized function with better or even equal nonlinearity. Direct implementation of F using truth table will take 2^{50} bits, which is not feasible to store. However, using the representation of F as (f, S_1, \ldots, S_{30}) , it is possible to implement F using 1 Megabit, i.e., 128 Kilobytes by implementing f by truth table. Moreover, if f is of the form $f(X_1, \ldots, X_{10}, Y_1, \ldots, Y_{10}) = 10$

 $\bigoplus_{i=1}^{10} X_i Y_i \oplus g(Y_1, \dots, Y_{10}) \oplus X_1 \dots X_{10} Y_1 \dots Y_{10}, \text{ implementation of } f \text{ requires}$ 1 Kilobit (128 bytes) of memory as we need to represent g by truth table only.

References

- P. Camion, C. Carlet, P. Charpin, and N. Sendrier. On correlation immune functions. In Advances in Cryptology CRYPTO'91, pages 86–100. Springer-Verlag, 1992.
- C. Carlet. More correlation immune and resilient functions over Galois fields and Galois rings. In Advances in Cryptology - EUROCRYPT'97, pages 422–433. Springer-Verlag, May 1997.
- 3. C. Carlet and P. Guillot. A characterization of bent functions. *Journal of Combinatorial Theory, Series A*, 76(2):328–335, September 1996.
- C. Ding, G. Xiao, and W. Shan. The Stability Theory of Stream Ciphers. Lecture Notes in Computer Science. Springer-Verlag, 1991.
- E. Filiol and C. Fontaine. Highly nonlinear balanced Boolean functions with a good correlation-immunity. In Advances in Cryptology - EUROCRYPT'98. Springer-Verlag, 1998.
- X. G. Zhen and J. Massey. A spectral characterization of correlation immune combining functions. *IEEE Transactions on Information Theory*, 34(3):569–571, May 1988.
- 7. R. W. Hamming. Coding And Information Theory. Prentice Hall Inc., 1980.
- S. Maitra and P. Sarkar. Enumeration of correlation immune Boolean functions. In 4th Australasian Conference on Information, Security and Privacy. Springer Verlag, Lecture Notes in Computer Science, No 1587, 7-9 April 1999.

- 9. J. Massey. Shift-Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory*, IT-15:122–127, January 1969.
- W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In Advances in Cryptology - EUROCRYPT'89, pages 549–562. Springer-Verlag, 1990.
- W. Millan, A. Clark, and E. Dawson. Heuristic design of cryptographically strong balanced Boolean functions. In Advances in Cryptology - EUROCRYPT'98. Springer-Verlag, 1998.
- C. J. Mitchell. Enumerating Boolean functions of cryptographic significance. *Journal of Cryptology*, 2(3):155–170, 1990.
- O. S. Rothaus. On bent functions. Journal of Combinatorial Theory, Series A20, pages 300–305, 1976.
- R. A. Rueppel and O. J. Staffelbach. Products of Linear Recurring Sequences with Maximum Complexity. *IEEE Transactions on Information Theory*, IT-33:124–131, January 1987.
- 15. J. Seberry, X. M. Zhang, and Y. Zheng. Nonlinearly balanced Boolean functions and their propagation characteristics. In *Advances in Cryptology CRYPTO'93*, pages 49–60. Springer-Verlag, 1994.
- J. Seberry, X. M. Zhang, and Y. Zheng. On constructions and nonlinearity of correlation immune Boolean functions. In Advances in Cryptology EUROCRYPT'93, pages 181–199. Springer-Verlag, 1994.
- T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-30(5):776–780, September 1984.
- 18. T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, C-34(1):81–85, January 1985.

UMAC: Fast and Secure Message Authentication

J. Black¹, S. Halevi², H. Krawczyk^{2,3}, T. Krovetz¹, and P. Rogaway¹

Abstract. We describe a message authentication algorithm, UMAC, which can authenticate messages (in software, on contemporary machines) roughly an order of magnitude faster than current practice (e.g., HMAC-SHA1), and about twice as fast as times previously reported for the universal hash-function family MMH. To achieve such speeds, UMAC uses a new universal hash-function family, NH, and a design which allows effective exploitation of SIMD parallelism. The "cryptographic" work of UMAC is done using standard primitives of the user's choice, such as a block cipher or cryptographic hash function; no new heuristic primitives are developed here. Instead, the security of UMAC is rigorously proven, in the sense of giving exact and quantitatively strong results which demonstrate an inability to forge UMAC-authenticated messages assuming an inability to break the underlying cryptographic primitive. Unlike conventional, inherently serial MACs, UMAC is parallelizable, and will have ever-faster implementation speeds as machines offer up increasing amounts of parallelism. We envision UMAC as a practical algorithm for next-generation message authentication.

1 Introduction

This paper describes a new message authentication code, UMAC, and the theory that lies behind it. UMAC has been designed with two main goals in mind: **extreme speed** and **provable security**. We aimed to create the fastest MAC ever described, and by a wide margin. (We are speaking of speed with respect to software implementations on contemporary general-purpose computers.) But we insisted that it be demonstrably secure, in the sense of having quantitatively desirable reductions from its underlying cryptographic primitives.

UMAC is certainly fast. On a 350 MHz Pentium II PC, one version of UMAC (where the adversary has 2^{-60} chance of forgery) gives a peak performance of 2.9 Gbits/sec (**0.98** cycles/byte). Another version of UMAC (with 2^{-30} chance of forgery) achieves peak performance of 5.6 Gbits/sec (**0.51** cycles/byte). For comparison, our SHA-1 implementation runs at **12.6** cycles/byte. (SHA-1 speed upper bounds the speed of HMAC-SHA1 [3], a software-oriented MAC representative of the speeds achieved by current practice.) The previous speed champion among proposed universal hash functions (the main ingredient for making a fast MAC; see below) was MMH [13], which runs at about **1.2** cycles/byte (for 2^{-30} chance of forgery) under its originally envisioned implementation.

 $^{^1\,}$ Department of Computer Science, University of California, Davis CA 95616 USA $^2\,$ IBM T.J. Watson Research Center, Yorktown Heights NY 10598 USA

³ Department of Electrical Engineering, Technion – Israel Institute of Technology, 32000 Haifa ISRAEL

How has it been possible to achieve these speeds? Interestingly, we have done this with the help of our second goal, provable security. We use the well-known universal-hashing approach to message authentication, introduced by [28], making innovations in its realization. Let us now review this approach and its advantages, and then describe what we have done to make it fly.

1.1 Universal-Hashing Approach

UNIVERSAL HASHING AND AUTHENTICATION. Our starting point is a universal hash-function family [10]. (Indeed the "U" in UMAC is meant to suggest the central role that <u>universal</u> hash-function families play in this MAC.) Remember that a set of hash functions is said to be " ϵ -universal" if for any pair of distinct messages, the probability that they collide (hash to the same value) is at most ϵ . The probability is over the random choice of hash function.

As described in [28], a universal hash-function family can be used to build a MAC. The parties share a secret and randomly chosen hash function from the hash-function family, and a secret encryption key. A message is authenticated by hashing it with the shared hash function and then encrypting the resulting hash. Wegman and Carter showed that when the hash-function family is strongly universal (a similar but stronger property than the one we defined) and the encryption is realized by a one-time pad, the adversary cannot forge with probability better than that obtained by choosing a random string for the MAC.

WHY UNIVERSAL HASHING? As suggested many times before [16,25,13], the above approach is a promising one for building a highly-secure and ultra-fast MAC. The reasoning is like this: the speed of a universal-hashing MAC depends on the speed of the hashing step and the speed of the encrypting step. But if the hash function compresses messages well (i.e., its output is short) then the encryption shouldn't take long simply because it is a short string that is being encrypted. On the other hand, since the combinatorial property of the universal hash-function family is mathematically proven (making no cryptographic hardness assumptions), it needs no "over-design" or "safety margin" the way a cryptographic primitive would. Quite the opposite: the hash-function family might as well be the fastest, simplest thing that one can prove universal.

Equally important, the above approach makes for desirable security properties. Since the cryptographic primitive is applied only to the (much shorter) hashed image of the message, we can select a cryptographically conservative design for this step and pay with only a minor impact on speed. And the fact that the underlying cryptographic primitive is used only on short and secret messages eliminates many avenues of attack. *Under this approach security and efficiency are not conflicting requirements—quite the contrary, they go hand in hand.*

QUEST FOR FAST UNIVERSAL HASHING. At least in principle, the universal-hashing paradigm has reduced the problem of fast message authentication to that of fast universal hashing. Thus there has been much work on the design of fast-to-compute universal hash-function families. Here is a glimpse of some of this work. Krawczyk [16] describes the "cryptographic CRC" which has very fast hardware

implementations and reasonably fast software implementations; it needs about 6 cycles/byte, as shown by Shoup [26]. Rogaway's "bucket hashing" [25] was the first hash-function family explicitly targeted for fast software implementation; it runs in about 1.5–2.5 cycles/byte. Halevi and Krawczyk devised MMH [13], which takes advantage of current CPU trends to hash at about 1.5–3 cycles/byte on modern CPUs.

With methods now in hand which hash so very quickly, one may ask if the hash-design phase of making a fast MAC is complete; after all, three cycles/byte may already be fast enough to keep up with high-speed network traffic. But authenticating information at the rate it is generated or transmitted is not the real goal; the goal is to use the smallest possible fraction of the CPU's cycles (so most of the machine's cycles are available for other work), by the simplest possible hash mechanism, and having the best proven bounds.

1.2 Our Contributions

Our work represents the next step in the quest for a practical, secure, high-speed MAC. Here we describe the main contributions associated to its design.

NEW HASH FUNCTION FAMILIES AND THEIR ANALYSES. A hash-function family named NH underlies hashing in UMAC. It is a simplification of the MMH and NMH families described in [13]. It works like this: the message M to hash is regarded as a sequence of an even number ℓ of integers, $M=(m_1,\ldots,m_\ell)$, where each $m_i \in \{0,\ldots,2^w-1\}$ corresponds to a w-bit word (e.g., w=16 or w=32). A particular hash function is named by a sequence of $n \geq \ell$ w-bit integers $K=(k_1,\ldots,k_n)$. We compute $\mathsf{NH}_K(M)$ as

$$\left(\sum_{i=1}^{\ell/2} \left((m_{2i-1} + k_{2i-1}) \bmod 2^w \right) \cdot \left((m_{2i} + k_{2i}) \bmod 2^w \right) \right) \bmod 2^{2w} . \tag{1}$$

The novelty of this method is that all the arithmetic is "arithmetic that computers like to do"—no finite fields or non-trivial modular reductions (as used in previous designs for universal hashing) come into the picture.

Despite the non-linearity of this hash function and despite its being defined using two different rings, $Z/2^w$ and $Z/2^{2w}$, not a finite field, we manage to obtain a tight bound on the collision probability: 2^{-w} (or 2^{-w+1} when using signed integers). Earlier analyses of related hash-function families had to give up a small constant in the analysis [13]. We give up nothing.

After proving our bounds on NH we extend the method using the "Toeplitz construction," a well-known approach to reduce the error probability without much lengthening of the key [18,16]. Prior to our work the Toeplitz construction was known to work only for *linear* functions over *fields*. Somewhat surprisingly, we prove that it also works for NH. Our proof again achieves a tight bound for the collision probability. We then make further extensions to handle length-issues, finally arriving at the hash-function family actually used in UMAC.

COMPLETE SPECIFICATION. Previous work on universal-hash-paradigm MACs dealt with fast hashing but did not address in detail how to embed a fast-to-

compute hash function into a concrete, practical, and fully-analyzed MAC. For some hash-function constructions (e.g., cryptographic CRCs) this step would be straightforward. But for the fastest hash families it is not, since these have some unpleasant characteristics, including specialized domains, long key-lengths, long output-lengths, or good performance only on very long messages. We show how to overcome such difficulties in a practical way which delivers on the promised speed. We provide a complete specification of UMAC, a ready-to-use MAC, in a separate specification document [7]. The technical difficulties mentioned above are not ignored; to the contrary, they are treated with the same care as the hash family itself. Our construction is fully analyzed, beginning-to-end; what is analyzed is exactly what is specified. This has only been possible by co-developing the specification document and the academic paper.

PRF(HASH, Nonce) Construction. Previous work has assumed that one hashes messages down to some fixed length string and then applies some cryptographic primitive. But using universal hashing to reduce a very long message to a fixed-length one can be complex, require long keys, or reduce the quantitative security. Instead, we reduce the length of the message by some pre-set constant factor, concatenate a sender-generated nonce, and then apply a pseudorandom function (PRF). See Section 5 for details.

EXPERIMENTS. We have been guided by extensive experimentation, through which we have identified the parameters that influence performance. While any reasonable setting of these parameters should out-perform conventional MACs, the fastest version of UMAC for one platform differs from the fastest version for another platform. We have therefore left UMAC parameterized, allowing specific choices to be fine-tuned to the application or platform at hand. Here and in [7] we consider a few reasonable settings for these parameters.

SIMD Exploitation. Unlike conventional, inherently serial MACs, UMAC is parallelizable, and will have ever-faster implementations as machines offer up increasing amounts of parallelism. Our algorithm and specification were specifically designed to exploit the form of parallelism offered by current and emerging SIMD architectures (Single Instruction Multiple Data). These provide some long registers that can, in certain instructions, be treated as vectors of smaller-sized words. For NH to run well we must quickly multiply w-bit numbers (w=16 or w=32) into their 2w-bit product. Many modern machines let us do this particularly well since we can re-appropriate instructions for vector dot-products that were primarily intended for multimedia applications. For now, our fastest implementation of UMAC runs on a Pentium and makes use of its MMX instructions which treat a 64-bit register as a vector of four 16-bit words.

1.3 Related Work

MMH PAPER. Halevi and Krawczyk investigated fast universal hashing in [13]. Their MMH construction takes advantage of improving CPU support for integer multiplication, particularly the ability to quickly multiply two 32-bit multiplicands into a 64-bit product. Their paper also describes a (formerly-unpublished)

hash-function family of Carter and Wegman, NMH^* , and a variation of it, NMH. Our NH function, as given by formula (1), is a simplification of NMH. The difference between NH and NMH is that NMH uses an additional modular reduction by a prime close to 2^w , followed by a reduction modulo 2^w . These changes simplify implementation, increase speed, and lower the collision probability.

OTHER WORK ON UNIVERSAL HASH MACS. Krawczyk describes a "cryptographic CRC" hash-function family. Shoup later studied the software performance of this construction, and gave several related ones [26]. In [16] one finds the Toeplitz construction used in a context similar to ours. An earlier use of the Toeplitz construction, in a different domain, can be found in [18]. A hash-function family specifically targeted for software was Rogaway's "bucket hashing" [25]. Its peak speed is fast but its long output length makes it suitable only for long messages. Nevelsteen and Preneel give a performance study of several universal hash functions proposed for MACs [19]. Patel and Ramzan give an MMH-variant that can be more efficient than MMH in certain settings [20]. Bernstein reports he has designed and implemented a polynomial-evaluation style hash-function family that runs in 4.5 Pentium cycles/byte [6]. Other recent work about universal hashing for authentication includes [1, 14].

OTHER TYPES OF MACS. One popular MAC is the CBC-MAC [2]. The CBC-MAC was analyzed by [5], and a variant of it was later analyzed in [21]. In [7] we make a PRF using a CBC-MAC variant similar to [21].

MACs have been constructed from cryptographic hash-functions. A few such methods are described in [27,15], and analysis appears in [22,23]. An increasingly popular MAC of this cryptographic-hash-function type is HMAC, which is described and analyzed in [3,12]. In one version of UMAC we suggest using HMAC as the underlying PRF. One can view UMAC as an alternative to HMAC, with UMAC being faster but more complex.

Full Version. The full version of this paper is in [8], and there is an associated specification document [7]. Reference code is also available [17].

2 Overview of UMAC

Unlike many MACs, our construction is stateful for the sender: when he wants to authenticate some string Msg he must provide as input to UMAC (along with Msg and the shared key Key) a 64-bit string Nonce. The sender must not reuse the nonce under the same MAC key. Typically the nonce would be a counter which the sender increments with each transmitted message.

The UMAC algorithm specifies how the message, key, and nonce determine an *authentication tag*. The sender will need to provide the receiver with the message, nonce, and tag. The receiver can then compute what "should be" the tag for this particular message and nonce, and see if it matches the received tag.

UMAC employs a *subkey generation process* in which the shared (convenient-length) key is mapped into UMAC's internal keys. In typical applications subkey generation is done just once, at the beginning of a communication session dur-

Subkey generation:

Using a PRG, map Key to $K = K_1 K_2 \cdots K_{1024}$, with each K_i a 32-bit word, and to A, where |A| = 512.

Hashing the message Msg to $HM = NHX_{Key}(Msg)$:

Let Len be $|Msg| \mod 4096$, encoded as a 2-byte string.

Append to Msg the minimum number of 0 bits to make |Msg| divisible by 8.

Let $\mathit{Msg} = \mathit{Msg}_1 \parallel \mathit{Msg}_2 \parallel \cdots \parallel \mathit{Msg}_t$ where each Msg_i is 1024 words

except for Msg_t , which has between 2 and 1024 words.

Let $HM = NH_K(Msg_1) \parallel NH_K(Msg_2) \parallel \cdots \parallel NH_K(Msg_t) \parallel Len$

Computing the authentication tag:

The tag is $Tag = HMAC-SHA1_A$ ($HM \parallel Nonce$)

Fig. 1. An illustrative special case of UMAC. The algorithm above computes a 160-bit tag given Key, Msg, and Nonce. See the accompanying text for the definition of NH.

ing which the key does not change, and so subkey-generation is usually not performance-critical.

UMAC depends on a few different *parameters*. We begin by giving a description of UMAC as specialized to one particular setting of these parameters. Then we briefly explore the role of various parameters.

2.1 An Illustrative Special Case

The underlying shared key Key (which might be, say, 128 bits) is first expanded into internal keys K and A, where K is 1024 words (a word being 32-bits) and A is 512 bits. How Key determines K and A is a rather standard detail (it can be handled using any PRG), and so we omit its description here. See Figure 1. There we refer to the hash function NH, which is applied to each block Msg_1, \ldots, Msg_t of Msg. We now define this function. Let $M = Msg_j$ be one of these blocks. Regard M as a sequence $M = M_1 \cdots M_\ell$ of 32-bit words, where $2 \le \ell \le 1024$. The hash function is named by $K = K_1 \cdots K_{1024}$, where K_i is 32 bits. We let $NH_K(M)$ be the 64-bit string $NH_K(M) =$

$$(M_1 +_{32} K_1) \times_{64} (M_2 +_{32} K_2) +_{64} \cdots +_{64} (M_{\ell-1} +_{32} K_{\ell-1}) \times_{64} (M_{\ell} +_{32} K_{\ell})$$

where $+_{32}$ is computer addition on 32-bit strings to give their 32-bit sum, $+_{64}$ is computer addition on 64-bit strings to give their 64-bit sum, and \times_{64} is computer multiplication on unsigned 32-bit strings to give their 64-bit product. This description of NH is identical to Equation (1) (for w = 32) but emphasizes that all the operations we are performing directly correspond to machine instructions of modern CPUs. See the left-hand side of Figure 2.

Theorem 1 says that NH is 2^{-32} -universal on strings of equal (and appropriate) length. Combining with Proposition 1 gives that NHX (as described in Figure 1) is 2^{-32} -universal, but now for any pair of strings. By Theorem 3, if an

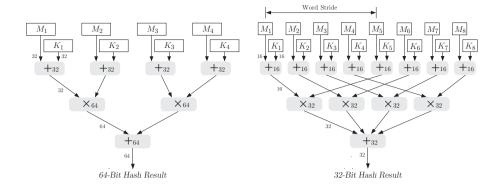


Fig. 2. Left: The NH hash function with a wordsize of w=32 bits, as used in UMAC-STD-30 and UMAC-STD-60. **Right:** The "strided" form of NH, with wordsize of w=16 bits, as used in UMAC-MMX-30 and UMAC-MMX-60.

adversary could forge a message with probability $2^{-32} + \delta$ then an adversary of essentially the same computational complexity could break HMAC-SHA1 (as a PRF) with advantage $\delta - 2^{-160}$. Under generally-accepted assumptions about SHA-1 only negligible values of δ can be achieved by a feasible attacker [3, 4], thus providing essentially a 2^{-32} upper bound on the forging probability against this version of UMAC.

2.2 UMAC Parameters

The full name of the version of NH just described is NH[n, w], where n = 1024 and w = 32: the wordsize is w = 32 bits and the blocksize is n = 1024 words. Numbers n and w are two of UMAC's parameters. Let us describe a few others.

Naturally enough, the *pseudorandom function* (PRF) which gets applied to $HM \parallel Nonce$ is a parameter. We used HMAC-SHA1 above, but any PRF is allowed. Similarly, a parameter specifies how Key gets mapped to K and A.

The universal hashing used in our example had collision probability 2^{-32} . We make provisions for lowering this. To square the collision probability one could hash the message twice, using independent hash keys, and concatenate the results. But an optimization in UMAC is that the two keys that are used are not independent; rather, one key is the "shift" of the other, with a few new words coming in. This is the well-known "Toeplitz construction." We prove in Theorem 2 that, for NH, the probability still drops according to the square.

In our example we used a long subkey K—it had 4096 bytes. To get good compression with a shorter subkey we can use two-level (2L) hashing. If a hash key of length n_1 gives compression ratio λ_1 and a hash key of length n_2 gives compression ratio λ_2 then using two levels of hashing gives compression ratio $\lambda_1\lambda_2$ with key size $\ell_1 + \ell_2$. Our specification allows for this. In fact, we allow 2L hashing in which the Toeplitz shift is applied at each level. It turns out that

this only loses a factor of two in the collision probability. The analysis is rather complex, and is omitted.

To accommodate SIMD architectures we allow slight adjustments in indexing. For example, to use the MMX instructions of the Pentium processor, instead of multiplying $(M_1 +_{16} K_1)$ by $(M_2 +_{16} K_2)$ and $(M_3 +_{16} K_3)$ by $(M_4 +_{16} K_4)$, we compute $(M_1 +_{16} K_1) \times_{32} (M_5 +_{16} K_5) +_{32} (M_2 +_{16} K_2) \times_{32} (M_6 +_{16} K_6) +_{32} \cdots$. There are MMX instructions which treat each of two 64-bit registers as four 16-bit words, corresponding words of which can be added or multiplied to give four 16-bit sums or four 32-bit products. So reading $M_1 \parallel M_2 \parallel M_3 \parallel M_4$ into one MMX register and $M_5 \parallel M_6 \parallel M_7 \parallel M_8$ into another we are well-positioned to multiply $M_1 +_{16} K_1$ by $M_5 +_{16} K_5$, not $M_2 +_{16} K_2$. See Figure 2.

There are a few more parameters. The sign parameter indicates whether the arithmetic operation \times_{64} is carried out thinking of the strings as unsigned (non-negative) or signed (twos-complement) integers. The MMX instructions mentioned above only operate on signed values, as does Java. If the input message is sufficiently short there is no speed savings to be had by hashing it with NH. The min-length-to-hash specifies the minimum-length message which should be hashed before being passed to the PRF. Finally, an endian parameter indicates if the MAC should favor big-endian or little-endian computation.

NAMED PARAMETER SETS. In [7] we suggest some settings for the vector of parameters, giving rise to UMAC-STD-30, UMAC-STD-60, UMAC-MMX-30, and UMAC-MMX-60. Here we summarize their salient features.

UMAC-STD-30 and UMAC-STD-60 use a wordsize of w=32 bits. They employ 2L hashing with a compression factor of 32 followed by a compression factor of 16. This corresponds to a subkey K of about 400 Bytes. They employ HMAC-SHA1 as the underlying PRF. They use signed arithmetic. The difference between UMAC-STD-30 and UMAC-STD-60 is the collision bound (and therefore forgery bound): 2^{-30} and 2^{-60} , respectively, which are achieved by hashing either once or twice (the latter using a Toeplitz-shifted key). These two versions of UMAC perform well on a wide range of contemporary processors.

UMAC-MMX-30 and UMAC-MMX-60 are well-suited to exploit the SIMD-parallelism available in the MMX instruction set of Intel processors. They use wordsize w=16 bits. Hashing is accomplished with a single-level scheme and a hash key of about 4 KBytes, which yields the same overall compression ratio as the 2L scheme used in the UMAC-STD variants. These MACs use the CBC-MAC of a software-efficient block cipher as the basis of the underlying PRF. Our tests were performed using the block cipher RC6 [24]. Arithmetic is again signed. The difference between UMAC-MMX-30 and UMAC-MMX-60 is the maximal forgery probability: 2^{-30} and 2^{-60} , respectively.

3 The NH Hash Family

Recall that NH is not, by itself, the hash-function family which UMAC uses, but the basic building block from which we construct UMAC's hash. After some brief preliminaries we define and analyze NH.

3.1 Preliminaries

FUNCTION FAMILIES. A family of functions (with domain $A \subseteq \{0,1\}^*$ and range $B \subseteq \{0,1\}^*$) is a set of functions $\mathsf{H} = \{h: A \to B\}$ endowed with some distribution. When we write $h \leftarrow \mathsf{H}$ we mean to choose a random function $h \in \mathsf{H}$ according to this distribution. A family of functions is also called a family of hash functions or a hash-function family.

Usually we specify a family of functions H by specifying some finite set of strings, Key, and explaining how each string $K \in \text{Key}$ names some function $H_K \in \text{H}$. We may then think of H not as a set of functions from A to B but as a single function $H : \text{Key} \times A \to B$, whose first argument we write as a subscript. A random element $h \in \text{H}$ is determined by selecting uniformly at random a string $K \in \text{Key}$ and setting $h = H_K$.

UNIVERSAL HASHING. We are interested in hash-function families in which "collisions" (when h(M) = h(M') for distinct M, M') are infrequent:

Definition 1. Let $H = \{h : A \to B\}$ be a family of hash functions and let $\epsilon \geq 0$ be a real number. We say that H is ϵ -universal, denoted ϵ -AU, if for all distinct $M, M' \in A$, we have that $\Pr_{h \leftarrow H}[h(M) = h(M')] \leq \epsilon$. We say that H is ϵ -universal on equal-length strings if for all distinct, equal-length strings $M, M' \in A$, we have that $\Pr_{h \leftarrow H}[h(M) = h(M')] \leq \epsilon$.

3.2 Definition of NH

Fix an even $n \geq 2$ (the "blocksize") and a number $w \geq 1$ (the "wordsize"). We define the family of functions $\mathsf{NH}[n,w]$ as follows. The domain is $A = \{0,1\}^{2w} \cup \{0,1\}^{4w} \cup \cdots \cup \{0,1\}^{nw}$ and the range is $B = \{0,1\}^{2w}$. Each function in $\mathsf{NH}[n,w]$ is named by an nw-bit string K; a random function in $\mathsf{NH}[n,w]$ is given by a random nw-bit string K. We write the function indicated by K as $\mathsf{NH}_K(\cdot)$.

Let U_w and U_{2w} represent the sets $\{0,\ldots,2^w-1\}$ and $\{0,\ldots,2^{2w}-1\}$, respectively. Arithmetic done modulo 2^w returns a result in U_w ; arithmetic done modulo 2^{2w} returns a result in U_{2w} . We overload the notation introduced in Section 2.1: for integers x,y let $(x+_w y)$ denote (x+y) mod 2^w . (Earlier this was an operator from strings to strings, but with analogous semantics.)

Let $M \in A$ and denote $M = M_1 \cdots M_\ell$, where $|M_1| = \cdots = |M_\ell| = w$. Similarly, let $K \in \{0,1\}^{nw}$ and denote $K = K_1 \cdots K_n$, where $|K_1| = \cdots = |K_n| = w$. Then $\mathsf{NH}_K(M)$ is defined as

$$\mathsf{NH}_K(M) \ = \ \sum_{i=1}^{\ell/2} \ (k_{2i-1} \ +_{^w} \ m_{2i-1}) \cdot (k_{2i} \ +_{^w} \ m_{2i}) \ \mathrm{mod} \ 2^{2w}$$

where $m_i \in U_w$ is the number that M_i represents (as an unsigned integer), where $k_i \in U_w$ is the number that K_i represents (as an unsigned integer), and the right-hand side of the above equation is understood to name the (unique) 2w-bit string which represents (as an unsigned integer) the U_{2w} -valued integer result. Henceforth we shall refrain from explicitly converting from strings to

integers and back, leaving this to the reader's good sense. (We comment that for everything we do, one could use any bijective map from $\{0,1\}^w$ to U_w , and any bijective map from U_{2w} to $\{0,1\}^{2w}$.) When the values of n and w are clear from the context, we write NH instead of NH[n,w].

3.3 Analysis

The following theorem bounds the collision probability of NH. In the full version of this paper [8] we also treat the signed case, yielding a bound of $2^{t(-w+1)}$ -AU.

Theorem 1. For any even $n \geq 2$ and $w \geq 1$, $\mathsf{NH}[n,w]$ is 2^{-w} -AU on equallength strings.

Proof. Let M, M' be distinct members of the domain A with |M| = |M'|. We are required to show $\Pr_{K \leftarrow \mathsf{NH}} [\mathsf{NH}_K(M) = \mathsf{NH}_K(M')] \leq 2^{-w}$. Converting the message and key strings to n-vectors of w-bit words we invoke the definition of NH to restate our goal as showing that

$$\Pr\left[\sum_{i=1}^{\ell/2} (k_{2i-1} + w m_{2i-1})(k_{2i} + w m_{2i}) = \sum_{i=1}^{\ell/2} (k_{2i-1} + w m'_{2i-1})(k_{2i} + w m'_{2i})\right]$$

is no more than 2^{-w} where the probability is taken over uniform choices of (k_1, \ldots, k_n) with each k_i in U_w . Above (and for the remainder of the proof) all arithmetic is carried out in $\mathbb{Z}/2^{2w}$.

Since M and M' are distinct, $m_i \neq m_i'$ for some $1 \leq i \leq n$. Since addition and multiplication in a ring are commutative, we lose no generality in assuming $m_2 \neq m_2'$. We now prove that for any choice of k_2, \dots, k_n we have $\Pr_{k_1 \in U_w} \left[(m_1 +_w k_1)(m_2 +_w k_2) + \sum_{i=2}^{\ell/2} (m_{2i-1} +_w k_{2i-1})(m_{2i} +_w k_{2i}) = (m_1' +_w k_1)(m_2' +_w k_2) + \sum_{i=2}^{\ell/2} (m_{2i-1}' +_w k_{2i-1})(m_2' +_w k_{2i}) \right] \leq 2^{-w}$, which will imply the theorem. Collecting up the summations, let

$$y = \sum_{i=2}^{\ell/2} (m_{2i-1} + w k_{2i-1})(m_{2i} + w k_{2i}) - \sum_{i=2}^{\ell/2} (m'_{2i-1} + w k_{2i-1})(m'_{2i} + w k_{2i})$$

and let $c = (m_2 +_w k_2)$ and $c' = (m'_2 +_w k_2)$. Note that c and c' are in U_w , and since $m_2 \neq m'_2$, we know $c \neq c'$. We rewrite the above probability as

$$\Pr_{k_1 \in U_w} \left[c(m_1 +_w k_1) - c'(m'_1 +_w k_1) + y = 0 \right] \le 2^{-w}.$$

In Lemma 1 below, we prove that there can be at most one k_1 in U_w satisfying $c(m_1 +_w k_1) - c'(m'_1 +_w k_1) + y = 0$, yielding the desired bound.

Lemma 1. Let c and c' be distinct values from U_w . Then for any $m, m' \in U_w$ and any $y \in U_{2w}$ there exists at most one $k \in U_w$ such that $c(k +_w m) = c'(k +_w m') + y$ in $\mathbb{Z}/2^{2w}$.

Proof. We note that it is sufficient to prove the case where m=0. We proceed to therefore prove that for any $c,c',m'\in U_w$ with $c\neq c'$ and any $y\in U_{2w}$ there is at most one $k\in U_w$ such that $kc=(k+_wm')c'+y$ in $Z/2^{2w}$. Since $k,m'<2^w$, we know that $(k+_wm')$ is either k+m' or $k+m'-2^w$, depending on whether $k+m'<2^w$ or $k+m'\geq 2^w$ respectively. So now we have

$$k(c - c') = m'c' + y \text{ and } k < 2^w - m'$$
 (2)

$$k(c-c') = (m'-2^w)c' + y \text{ and } k \ge 2^w - m'$$
 (3)

Lemma 2, presented next, shows that there is at most one solution to each of the equations above. The remainder of the proof is devoted to showing there cannot exist $k = k_1 \in U_w$ satisfying (2) and $k = k_2 \in U_w$ satisfying (3) in $\mathbb{Z}/2^{2w}$. Suppose such a k_1 and k_2 did exist. Then we have $k_1 < 2^w - m'$ with $k_1(c-c') = m'c' + y$ and $k_2 \geq 2^w - m'$ with $k_2(c-c') = (m'-2^w)c' + y$. Subtracting the former from the latter yields $(k_2 - k_1)(c' - c) = 2^wc'$. We show that this equation has no solutions in $\mathbb{Z}/2^{2w}$. There are cases:

CASE 1: c' > c. Since both $(k_2 - k_1)$ and (c' - c) are positive and smaller than 2^w , their product is also positive and smaller than 2^{2w} . And since $2^wc'$ is also positive and smaller than 2^{2w} , it is sufficient to show that the equation has no solutions in Z. But this is clear, since $(k_2 - k_1) < 2^w$ and $(c' - c) \le c'$, and so necessarily $(k_2 - k_1)(c' - c) < 2^wc'$.

CASE 2: c' < c. Here we show $(k_2 - k_1)(c - c') = -2^w c'$ has no solutions in $\mathbb{Z}/2^{2w}$. As before, we convert to \mathbb{Z} , to yield $(k_2 - k_1)(c - c') = 2^{2w} - 2^w c'$. But again $(k_2 - k_1) < 2^w$ and $(c - c') < (2^w - c')$, so $(k_2 - k_1)(c - c') < 2^w(2^w - c') = 2^{2w} - 2^w c'$.

Let $D_w = \{-2^w + 1, \dots, 2^w - 1\}$ be the values attainable from a difference of any two elements of U_w . The following lemma completes the proof.

Lemma 2. Let $x \in D_w$ be nonzero. Then for any $y \in U_{2w}$, there exists at most one $a \in U_w$ such that ax = y in $\mathbb{Z}/2^{2w}$.

Proof. Suppose there were two distinct elements $a, a' \in U_w$ such that ax = y and a'x = y. Then ax = a'x so x(a - a') = 0. Since x is nonzero and a and a' are distinct, the foregoing product is $2^{2w}k$ for nonzero k. But x and (a - a') are in D_w , and therefore their product is in $\{-2^{2w} + 2^{w+1} - 1, \dots, 2^{2w} - 2^{w+1} + 1\}$, which contains no multiples of 2^{2w} other than 0.

COMMENTS. The above bound is tight; let $M = 0^w 0^w$ and $M' = 1^w 0^w$ and note that any key $K = K_1 K_2$ with $K_2 = 0^w$ causes a collision.

Although we do not require any stronger properties than the above, NH is actually 2^{-w} -A Δ U under the operation of addition modulo 2^{2w} . Only trivial modifications to the above proof are required. See [13] for a definition of ϵ -A Δ U.

Several variants of NH fail to preserve collision probability $\epsilon = 2^{-w}$. In particular, replacing the inner addition or the outer addition with bitwise-XOR increases ϵ substantially. However, removing the inner moduli retains $\epsilon = 2^{-w}$ (but significantly degrades performance).

There is also a version of $\mathsf{NH}[n,w]$, called $\mathsf{NHS}[n,w]$, that uses signed arithmetic. Surprisingly, the signed version of NH has slightly higher collision probability: the full paper proves a tight bound of 2^{-w+1} -AU. This helps explain the 2^{-30} and 2^{-60} forging probabilities for the four UMAC versions named in Section 2.2 and performance-measured in Section 6; all four algorithms use the signed version of NH .

4 Extending NH

The hash-function family NH is not yet suitable for use as a MAC: for one thing, it operates only on strings of "convenient" lengths (ℓw -bit strings for even $\ell \leq n$). Also, its collision probability may be higher than desired (2^{-w} when one may want 2^{-2w} or 2^{-4w}), and this is guaranteed only for strings of equal length. We remedy these deficiencies in this section.

4.1 Reducing Collision Probability: NH-Toeplitz

Should we wish to reduce the collision probability for NH, we have a few options. Increasing the wordsize w yields an improvement, but architectural characteristics dictate the natural values for w. Another well-known technique is to apply several random members of our hash-function family to the message, and concatenate the results. If we concatenate the results from, say, four independent instances of the hash function, the collision probability drops from 2^{-w} to 2^{-4w} . However this solution requires four times as much key material. A superior (and well-known) idea is to use the Toeplitz-extension of our hash-function families: given one key we "left shift" to get the "next" key and hash again. For example, to reduce the collision probability to 2^{-64} for NH[n, 16], we choose a single key $K = (K_1, \ldots, K_{n+6})$ and hash with the four derived keys $(K_{1+2i}, \ldots, K_{n+2i})$ where $0 \le i \le 3$. This trick not only saves key material, but it can also improve performance by reducing memory accesses, increasing locality of memory references and increasing parallelism.

Since these keys are related, it is not clear that the collision probability indeed drops to the desired value of 2^{-64} . Although there are established results which yield this bound (e.g., [18]), they only apply to linear hashing schemes over fields. Instead, NH is non-linear and operates over a combination of rings $(Z/2^w)$ and $Z/2^{2w}$. In Theorem 2 we prove that the Toeplitz construction nonetheless achieves the desired bound in the case of NH.

We define the hash-function family $\mathsf{NH}^\mathsf{T}[n,w,t]$ ("Toeplitz-NH") as follows. Fix an even $n \geq 2$, $w \geq 1$, and $t \geq 1$ (the "Toeplitz iteration count"). The domain $A = \{0,1\}^{2w} \cup \{0,1\}^{4w} \cup \cdots \cup \{0,1\}^{nw}$ remains as it was for NH, but the range is now $B = \{0,1\}^{2wt}$. A function in $\mathsf{NH}^\mathsf{T}[n,w,t]$ is named by a string K of w(n+2(t-1)) bits. Let $K = K_1 \parallel \cdots \parallel K_{n+2(t-1)}$ (where each K_i is a w-bit word), and let the notation $K_{i...j}$ represent $K_i \parallel \cdots \parallel K_j$. Then for any $M \in A$ we define $\mathsf{NH}^\mathsf{T}_K(M)$ as

$$\mathsf{NH}_K^\mathsf{T}(M) = \mathsf{NH}_{K_{1..n}}(M) \parallel \mathsf{NH}_{K_{3..n+2}}(M) \parallel \cdots \parallel \mathsf{NH}_{K_{(2t-1),(n+2t-2)}}(M).$$

When clear from context we write NH^T instead of $NH^T[n, w, t]$.

The following shows that NH^T enjoys the best bound that one could hope for. The proof is in the full version [8].

Theorem 2. For any $w, t \ge 1$ and any even $n \ge 2$, $NH^T[n, w, t]$ is 2^{-wt} -AU on equal-length strings.

4.2 Padding, Concatenation, and Length Annotation

With NH^T we can decrease the collision probability to any desired level but we still face the problem that this function operates only on strings of "convenient" length, and that it guarantees this low collision probability only for equal-length strings. We solve these problems in a generic manner, with a combination of padding, concatenation, and length annotation.

MECHANISM. Let $\mathsf{H}:\{A\to B\}$ be a family of hash functions where functions in H are defined only for particular input lengths, up to some maximum, and all the hash functions have a fixed output length. Formally, the domain is $A=\bigcup_{i\in I}\{0,1\}^i$ for some finite nonempty index set $I\subseteq \mathsf{N}$ and the range is $B=\{0,1\}^\beta$, where β is some positive integer. Let a (the "blocksize") be the length of the longest string in A and let $\alpha \geq \lceil \lg_2 a \rceil$ be large enough to describe $|M| \mod a$. Then we define $\mathsf{H}^*=\{h^*:\{0,1\}^*\to\{0,1\}^*\}$ as follows.

function $h^*(Msg)$

- 1. if $M = \lambda$ then return 0^{α}
- 2. View Msg as a sequence of "blocks", $Msg = Msg_1 \parallel \cdots \parallel Msg_t$, with $|Msg_j| = a$ for all $1 \leq j < t$, and $1 \leq |Msg_t| \leq a$
- 3. Let Len be an α -bit string that encodes $|Msg| \mod a$
- 4. Let $i \geq 0$ be the least number such that $Msg_t \parallel 0^i \in A$
- 5. $Msg_t = Msg_t \parallel 0^i$
- 6. return $h(Msg_1) \parallel \cdots \parallel h(Msg_t) \parallel Len$

ANALYSIS. The following proposition indicates that we have correctly extended H to H*. The straightforward proof is in the full paper [8].

Proposition 1. Let $I \subseteq \mathbb{N}$ be a nonempty finite set, let $\beta \geq 1$ be a number, and let $H = \{h : \bigcup_{i \in I} \{0,1\}^i \to \{0,1\}^\beta\}$ be a family of hash functions. Let $H^* = \{h^* : \{0,1\}^* \to \{0,1\}^*\}$ be the family of hash functions obtained from H as described above. Suppose H is ϵ -AU on strings of equal length. Then H^* is ϵ -AU (across all strings).

5 From Hash to MAC

In this section we describe a way to make a secure MAC from an ϵ -AU family of hash functions (with small ϵ) and a secure pseudorandom function (PRF).

DEFINITION OF THE PRF(HASH, NONCE) CONSTRUCTION. We use a family of (hash) functions $\mathsf{H} = \{h : \{0,1\}^* \to \{0,1\}^*\}$ and a family of (random or pseudorandom) functions $\mathsf{F} = \{f : \{0,1\}^* \to \{0,1\}^{\tau}\}$. These are parameters of the construction. We also fix a set $\mathsf{Nonce} = \{0,1\}^{\tau}\}$ and an "encoding scheme" $\langle \cdot, \cdot \rangle$. The encoding scheme is a linear-time computable function that maps a string $HM \in \{0,1\}^*$ and $Nonce \in \mathsf{Nonce}$ into a string $\langle HM, Nonce \rangle$ of length |HM| + |Nonce| + O(1) from which, again in linear time, one can recover HM and Nonce. The MAC scheme $\mathsf{UMAC}[\mathsf{H},\mathsf{F}] = (\mathsf{KEY},\mathsf{TAG})$ is defined as:

function KEY() function
$$\mathrm{TAG}_{(f,h)}$$
 (M , $Nonce$) $f \leftarrow \mathsf{F}; h \leftarrow \mathsf{H}$ return (f,h) return $f(\langle h(M), Nonce \rangle)$

The keyspace for this MAC is $Key = H \times F$; that is, a random key for the MAC is a random hash function $h \in H$ together with a random function $f \in F$. Here we have regarded a MAC scheme as a pair consisting of a key-generation algorithm and a tag-generation algorithm. The formalization is in [8].

We point out that the use of the nonce does not, by itself, address the question of replay detection. Our definition of MAC security [8] speaks to an adversary's inability to produce a new (M, Nonce, Tag) tuple, but is silent on the question of when the verifier should regard an (M, Nonce, Tag) tuple as valid. Certainly Tag should be the correct tag for the given message and nonce, but the verifier may demand more. In particular, the verifier may wish to reject messages for which the nonce was used before. If so, replay attacks will be thwarted. Of course the verifier will need to maintain state to do this. If the sender and receiver use a counter for detecting replays this same counter can be UMAC's nonce.

ANALYSIS. We begin with the information-theoretic version of the scheme. The proof and relevant definitions are in the full paper [8]. Here it suffices to indicate that $\mathsf{Succ}^{\mathsf{mac}}_{\Sigma}(F)$ measures the chance that F forges under the MAC scheme Σ .

Lemma 3. Let $\epsilon \geq 0$ be a real number and let $H = \{h : \{0,1\}^* \rightarrow \{0,1\}^*\}$ be an ϵ -AU family of hash functions. Let $\tau \geq 1$ be a number and let $\Sigma = \mathsf{UMAC}[\mathsf{H},\mathsf{Rand}(\tau)]$ be the MAC scheme described above. Then for every adversary F we have that $\mathsf{Succ}^{\mathrm{mac}}_{\Sigma}(F) \leq \epsilon + 2^{-\tau}$.

In the usual way we can extend the above information-theoretic result to the complexity-theoretic setting. Roughly, we prove that if the hash-function family is ϵ -AU and no reasonable adversary can distinguish the PRF from a truly random function with advantage exceeding δ then no reasonable adversary can break the resulting MAC scheme with probability exceeding $\epsilon + \delta$.

The theorem refers to $\mathsf{Succ}^{\mathsf{mac}}_\Sigma(t,q,\mu)$, which is the maximal chance of forging by an adversary that runs in time t and asks q queries, these totaling μ bits. And it refers to $\mathsf{Adv}^{\mathsf{prf}}_\mathsf{F}(t',q',\mu')$, which measures the maximal possible advantage in distinguishing F from a random function among adversaries that run in time t' and asks q' queries, these totaling μ' bits. If H is a family of hash functions then Time_H is an amount of time adequate to compute a representation for a random $h \leftarrow \mathsf{H}$, while $\mathsf{Time}_h(\mu)$ is an amount of time adequate to evaluate h on strings whose lengths total μ bits. The proof of the following is standard.

Theorem 3. Let $\epsilon \geq 0$ be a real number, let $H = \{h : \{0,1\}^* \rightarrow \{0,1\}^*\}$ be an ϵ -AU family of hash functions, let $\tau \geq 1$ be a number and let $F : \{0,1\}^{\alpha} \times \{0,1\}^* \rightarrow \{0,1\}^{\tau}$ be a PRF. Let $\Sigma = \mathsf{UMAC}[\mathsf{H},\mathsf{F}]$ be the MAC scheme described above. Then

$$\mathsf{Succ}^{\mathrm{mac}}_{\varSigma}(t,q,\mu) \ \leq \ \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(t',q',\mu') + \epsilon + 2^{-\tau}$$

where
$$t' = t + \mathsf{Time}_{\mathsf{H}} + \mathsf{Time}_{h}(\mu) + O(\mu)$$
 and $q' = q + 1$ and $\mu' = \mu + O(q)$.

DISCUSSION. The use of the nonce is important for getting a quantitatively desirable security bound. Let $\overline{\mathsf{UMAC}}[\mathsf{H},\mathsf{F}]$ be the scheme which is the same as UMAC , except that the PRF is applied directly to HM , rather than to $\langle \mathit{HM}, \mathit{Nonce} \rangle$. Then the analog of Lemma 3 would say: fix a number $\tau \geq 1$, let $\mathsf{H} = \{h: \{0,1\}^* \to \{0,1\}^*\}$ be an ϵ -AU family of hash functions, and let $\varSigma = \overline{\mathsf{UMAC}}[\mathsf{H}, \mathsf{Rand}(\tau)]$. Then for every adversary F, $\mathsf{Succ}_{\varSigma}^{\mathsf{mac}}(F) \leq q^2 \epsilon + 2^{-\tau}$. This is a far cry from our earlier bound of $\epsilon + 2^{-\tau}$. And the problem is not with the analysis, but with the scheme itself: if one asks $\epsilon^{-1/2}$ oracle queries of $\overline{\mathsf{UMAC}}$ then, by the birthday bound, there is indeed a good chance to find distinct messages, M_1 and M_2 , which yield the same authentication tag. This is crucial information about the hash function which has now been leaked.

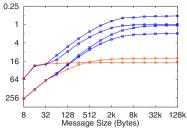
Compared to the original suggestion of [28], where one encrypts the hash of the message by XOR-ing with a one-time pad, we require a weaker assumption about the hash-function family H: it need only be ϵ -AU. (The original approach of [28] needs of H the stronger property of being "XOR almost-universal" [16].) Furthermore, it is no problem for us that the range of $h \in H$ has strings of unbounded length, while the hash functions used for [28] should have fixed-length output. On the other hand, our cryptographic tool is potentially stronger than what the complexity-theoretic version of [28] requires: we need a PRF over the domain Encoding of possible $\langle HM, Nonce \rangle$ encodings.

Compared to the PRF(HASH, Nonce) method, PRF(Nonce) \oplus HASH does have some advantages. One would need a final layer of almost-XOR-universal hashing, like a CRC hash [16, 26], or an NMH hash [13] with a conceptually infinite (PRG-generated) key. We are still investigating such possibilities.

6 Performance

We implemented the four flavors of UMAC named in Section 2.2 on three different platforms: a 350 MHz Intel Pentium II, a 200 MHz IBM/Motorola PowerPC 604e, and a 300 MHz DEC Alpha 21164. The code was written mostly in C, with a few functions done in assembly. For the Pentium II we wrote assembly for RC6, SHA-1, and the first-level NH hashes. For the PowerPC we did assembly for just the first-level NH hashes. In both cases, the number of lines of assembly written was small: about 80 lines.

For each combination of options we determined the scheme's throughput on variously sized messages, eight bytes through 512 KBytes. The experimental setup ensured that messages resided in level-1 cache regardless of their length.



	Pentium II		PowerPC		Alpha	
UMAC-STD-30	2.79	(1.03)	2.28	(1.26)	1.79	(1.60)
UMAC-STD-60	1.49	(1.93)	1.81	(1.58)	1.03	(2.78)
UMAC-MMX-30	5.66	(0.51)	1.19	(2.40)	0.571	(5.02)
UMAC-MMX-60	2.94	(0.98)	0.684	(4.19)	0.287	(10.0)
CBC-MAC-RC6	0.162	(17.7)	0.210	(13.7)	0.068	(42.5)
HMAC-SHA1	0.227	(12.6)	0.228	(12.6)	0.117	(24.5)

Fig. 3. UMAC Performance. Left: Performance over various message lengths on a Pentium II, measured in machine cycles/byte. The lines in the graph correspond to the following MACs (beginning at the top-right and moving downward): UMAC-MMX-30, UMAC-MMX-60, UMAC-STD-30, UMAC-STD-60, HMAC-SHA1 and CBC-MAC-RC6. Right: Peak performance for three platforms, measured in Gbits/sec (cycles/byte). The Gbits/sec numbers are normalized to 350 MHz.

For comparison the same tests were run for HMAC-SHA1 [11] and the CBC-MAC of a fast block cipher, RC6 [24].

The graph in Figure 3 shows the throughput of the four versions of UMAC, as well as HMAC-SHA1 and CBC-MAC-RC6, all run on our Pentium II. The table gives peak throughput for the same MACs, but on all three platforms. The performance curves for the Alpha and PowerPC look similar to the Pentium II—they perform better than the reference MACs at around the same message length, and level out at around the same message length.

As the data shows, the MMX versions are much faster than the STD versions on the Pentium. Going from words of w=32 bits to w=16 bits might appear to increase the amount of work needed to get to a given collision bound, but a single MMX instruction can do four 16-bit multiplications and two 32-bit additions. This is more work per instruction than the corresponding 32-bit instructions.

UMAC-STD uses only one-tenth as much hash key as UMAC-MMX to achieve the same compression ratio. The penalty for such 2L hashing ranges from 8% on small messages to 15% on long ones. To lower the amount of key material we could have used a one-level hash with a smaller compression ratio, but experiments show this is much less effective: relative to UMAC-MMX-60, which uses about 4 KBytes of hash key, a 2 KBytes scheme goes 85% as fast, a 1 KByte scheme goes 66% as fast, and a 512 bytes scheme goes 47% as fast.

Another experiment replaced the NH hash function used in UMAC-STD-30 by MMH [13]. Peak performance dropped by 24%. We replaced the NH hash function of UMAC-MMX-30 by a 16-bit MMH and performance dropped by 5%.

We made a 2^{-15} -forgery probability UMAC-MMX-15 in the natural way, which ran in 0.32 cycles/bytes on our Pentium II.

We tried UMAC-STD-30 and UMAC-STD-60 on a Pentium processor which lacked MMX. Peak speeds were 2.2 cycles/byte and 4.3 cycles/byte—still well ahead of methods like HMAC-SHA1.

7 Directions

An interesting possibility (suggested to us by researchers at NAI Labs—see acknowledgments) is to restructure UMAC so that a receiver can verify a tag to various forgery probabilities—e.g., changing UMAC-MMX-60 to allow tags to be verified, at increasing cost, to forging probabilities of 2^{-15} , 2^{-30} , 2^{-45} , or 2^{-60} . Such a feature is particularly attractive for authenticating broadcasts to receivers of different security policies or computational capabilities.

Acknowledgments

Thanks to Mihir Bellare and the CRYPTO '99 Committee for their suggestions. Thanks to Dave Balenson and Dave Carman (of NAI Labs, the security research division of Network Associates; DARPA F30602-98-C-0215) for the idea mentioned in Section 7. Thanks to them, and to Bill Aiello, for pointing out the usefulness of high-forgery-probability MACs, such as UMAC-MMX-15, to applications like telephony and multimedia.

Rogaway, Black, and Krovetz were supported by Rogaway's NSF CAREER Award CCR-962540, and by MICRO grants 97-150 and 98-129, funded by RSA Data Security, Inc., and ORINCON Corporation. Much of Rogaway's work was carried out while on sabbatical at Chiang Mai University, under Prof. Krisorn Jittorntrum and Prof. Darunee Smawatakul.

References

- AFANASSIEV, V., GEHRMANN, C., AND SMEETS, B. Fast message authentication using efficient polynomial evaluation. In *Proceedings of the 4th Workshop on Fast Software Encryption* (1997), vol. 1267, Springer-Verlag, pp. 190–204.
- Ansi X9.9. American national standard Financial institution message authentication (wholesale). ASC X9 Secretariat American Bankers Association, 1986.
- BELLARE, M., CANETTI, R., AND KRAWCZYK, H. Keying hash functions for message authentication. In Advances in Cryptology CRYPTO '96 (1996), vol. 1109 of Lecture Notes in Computer Science, Springer-Verlag, pp. 1–15.
- BELLARE, M., CANETTI, R., AND KRAWCZYK, H. Pseudorandom functions revisited: The cascade construction. In 37th Annual Symposium on Foundations of Computer Science (1996), IEEE Computer Society, pp. 514–523.
- BELLARE, M., KILIAN, J., AND ROGAWAY, P. The security of cipher block chaining. In Advances in Cryptology CRYPTO '94 (1994), vol. 839 of Lecture Notes in Computer Science, Springer-Verlag, pp. 341–358.
- BERNSTEIN, D. Guaranteed message authentication faster than MD5. Unpublished manuscript, 1999.
- BLACK, J., HALEVI, S., HEVIA, A., KRAWCZYK, H., KROVETZ, T., AND ROG-AWAY, P. UMAC — Message authentication code using universal hashing. Unpublished specification, www.cs.ucdavis.edu/~rogaway/umac, 1999.
- 8. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and Rogaway, P. UMAC: Fast and secure message authentication. In *Advances in Cryptology CRYPTO '99* (1999), Lecture Notes in Computer Science, Springer-Verlag. Full version of this paper, available at www.cs.ucdavis.edu/~rogaway/umac.

- BRASSARD, G. On computationally secure authentication tags requiring short secret shared keys. In Advances in Cryptology – CRYPTO '82 (1983), Springer-Verlag, pp. 79–86.
- CARTER, L., AND WEGMAN, M. Universal hash functions. J. of Computer and System Sciences, 18 (1979), 143–154.
- 11. FIPS 180-1. Secure hash standard. NIST, US Dept. of Commerce, 1995.
- H. KRAWCZYK, M. B., AND CANETTI, R. HMAC: Keyed hashing for message authentication. IETF RFC-2104, 1997.
- HALEVI, S., AND KRAWCZYK, H. MMH: Software message authentication in the Gbit/second rates. In *Proceedings of the 4th Workshop on Fast Software Encryption* (1997), vol. 1267, Springer-Verlag, pp. 172–189.
- JOHANSSON, T. Bucket hashing with small key size. In Advances in Cryptology EUROCRYPT '97 (1997), Lecture Notes in Computer Science, Springer-Verlag.
- Kaliski, B., and Robshaw, M. Message authentication with MD5, 1995. Technical newsletter of RSA Laboratories.
- 16. Krawczyk, H. LFSR-based hashing and authentication. In *Advances in Cryptology CRYPTO '94* (1994), vol. 839 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 129–139.
- 17. KROVETZ, T. UMAC reference code (in ANSI C with Pentium assembly). Available from www.cs.ucdavis.edu/~rogaway/umac, 1999.
- 18. Mansour, Y., Nissan, N., and Tiwari, P. The computational complexity of universal hashing. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing* (1990), ACM Press, pp. 235–243.
- 19. NEVELSTEEN, W., AND PRENEEL, B. Software performance of universal hash functions. In *Advances in Cryptology EUROCRYPT '99* (1999), vol. 1592 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 24–41.
- PATEL, S., AND RAMZAN, Z. Square hash: Fast message authentication via optimized universal hash functions. In Advances in Cryptology CRYPTO '99 (1999), Lecture Notes in Computer Science, Springer-Verlag.
- 21. Petrank, E., and Rackoff, C. CBC MAC for real-time data sources. Manuscript 97-10 in http://philby.ucsd.edu/cryptolib.html, 1997.
- PRENEEL, B., AND VAN OORSCHOT, P. MDx-MAC and building fast MACs from hash functions. In Advances in Cryptology — CRYPTO '95 (1995), vol. 963 of Lecture Notes in Computer Science, Springer-Verlag, pp. 1–14.
- 23. PRENEEL, B., AND VAN OORSCHOT, P. On the security of two MAC algorithms. In *Advances in Cryptology EUROCRYPT '96* (1996), vol. 1070 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 19–32.
- RIVEST, R., ROBSHAW, M., SIDNEY, R., AND YIN, Y. The RC6 block cipher.
 Available from http://theory.lcs.mit.edu/~rivest/publications.html, 1998.
- ROGAWAY, P. Bucket hashing and its application to fast message authentication. In Advances in Cryptology - CRYPTO '95 (1995), vol. 963 of Lecture Notes in Computer Science, Springer-Verlag, pp. 313-328.
- Shoup, V. On fast and provably secure message authentication based on universal hashing. In Advances in Cryptology – CRYPTO '96 (1996), vol. 1109 of Lecture Notes in Computer Science, Springer-Verlag, pp. 74–85.
- TSUDIK, G. Message authentication with one-way hash functions. In *Proceedings of Infocom '92* (1992), IEEE Press.
- 28. Wegman, M., and Carter, L. New hash functions and their use in authentication and set equality. In *J. of Comp. and System Sciences* (1981), vol. 22, pp. 265–279.

Square Hash: Fast Message Authentication via Optimized Universal Hash Functions*

Mark Etzel¹, Sarvar Patel¹, and Zulfikar Ramzan^{⋆⋆2}

1 Bell Labs, Lucent Technologies
{mhetzel,sarvar}@bell-labs.com
2 Laboratory for Computer Science, MIT
 zulfikar@theory.lcs.mit.edu

Abstract. This paper introduces two new ideas in the construction of fast universal hash functions geared towards the task of message authentication. First, we describe a simple but novel family of universal hash functions that is more efficient than many standard constructions. We compare our hash functions to the MMH family studied by Halevi and Krawczyk [12]. All the main techniques used to optimize MMH work on our hash functions as well. Second, we introduce additional techniques for speeding up our constructions; these techniques apply to MMH and may apply to other hash functions. The techniques involve ignoring certain parts of the computation, while still retaining the necessary statistical properties for secure message authentication. Finally, we give implementation results on an ARM processor. Our constructions are general and can be used in any setting where universal hash functions are needed; therefore they may be of independent interest.

Key words: Message authentication codes, Universal Hashing.

1 Introduction

MESSAGE AUTHENTICATION. Designing good Message Authentication schemes is a very important objective in cryptography. The goal in message authentication is for one party to efficiently transmit a message to another party in such a way that the receiving party can determine whether or not the message he receives has been tampered with. The setting involves two parties, Alice and Bob, who have agreed on a pre-specified secret key x. Two algorithms are used: an algorithm S_x that applies a tag to a message, and a verification algorithm V_x that checks if the tag associated with a given message is valid. If Alice wants to send a message M to Bob, she first computes a message authentication code, or MAC, $\mu = S_x(M)$. She sends (M, μ) to Bob, and upon receiving the pair, Bob computes $V_x(M, \mu)$ which returns 1 if the MAC is valid, or returns 0 otherwise. Without knowledge of the secret key x, it should be infeasible for an adversary

^{*} Extended abstract. A full version is available at http://theory.lcs.mit.edu/~zulfikar

^{**} Work done while this author was at Lucent Technologies. This author would like to acknowledge DARPA grant DABT63-96-C-0018 and an NSF graduate fellowship.

to construct a message and a corresponding MAC that the verification algorithm will accept as valid. The formal security requirement for a MAC was first defined by Bellare, et al [4]. This definition is analogous to the formal security definition of a digital signature [11]. In particular, we say that an adversary forges a MAC if, when given oracle access to (S_x, V_x) , where x is kept secret, the adversary can come up with a pair (M^*, μ^*) such that $V_x(M^*, \mu^*) = 1$ but the message M^* was never made an input to the oracle for S_x .

IMPORTANCE OF EFFICIENT MACS. In general, MACs are computed frequently and on inputs which are often thousands of bytes long. Moreover, computing and verifying tags is typically done in software, and may be done on relatively weak platforms. Additionally, the computations must be done in real time. Therefore, developing techniques for optimizing MAC Algorithms while retaining the appropriate level of security is crucial. This paper presents two novel ideas in this direction.

COMMON APPROACHES TO MESSAGE AUTHENTICATION. One approach to message authentication involves using a secure block cipher, such as DES [21], in cipher block chaining mode. Another approach to message authentication, often seen in practice, involves using cryptographic hash functions like MD5 [24]. For example, one approach was to set $\mu = MD5(x \cdot m \cdot x)$; unfortunately, this particular scheme is vulnerable to a clever key recovery attack due to Preneel and and van Oorschot [23]. Other work on using cryptographic hash functions in MACs is the HMAC construction of Bellare, et al [3]; their schemes are good because they use fast and secure cryptographic building blocks. At first it appears that these techniques yield the most efficient results; however, Wegman and Carter [28] discovered that universal hash functions, allow us to avoid using heavy duty cryptographic primitives on the entire input string.

THE UNIVERSAL HASH FUNCTION APPROACH. In this approach, one starts with a family of hash functions H where for any pair $m \neq m'$ and for any element δ in the range, $\Pr_h[h(m) - h(m') = \delta] \leq \epsilon$. Here ϵ is a parameter related to the security of the MAC). Such functions are called ϵ - Δ -universal hash functions. Now, in order to compute the authentication tag for a message m, the communicating parties secretly agree on a function $h \in H$ chosen at random, and on a sequence of random pads $p_1, p_2 \dots$ To compute a MAC on the *i*-th message m_i , the sender computes $\mu_i = h(m_i) + p_i$. One remarkable aspect of this approach is that, even if a computationally unbounded adversary performs q black-box oracle queries to both algorithms used by the MAC, he has probability less than $q\epsilon$ to forge the MAC. The idea in the Wegman-Carter construction is to pre-process a message quickly using universal hash functions, and then apply a cryptographic operation such as a one-time pad. In general, the one-time pad can be replaced by pseudo-random sequence. Then, the parties would have to pre-agree on the function h and on the seed s which would be fed to either a pseudo-random generator or a pseudo-random function [10]. This approach to message authentication was first studied in [6]. If pseudo-randomness is used, then the resulting MAC is secure against a polynomially bounded adversary.

THE SQUARE HASH. This paper introduces two new ideas in the construction of fast universal hash functions. We start with a simple but novel family of universal hash functions which should be more efficient than certain well-known hash function families. The efficiency lies in the fact that whereas other common constructions involve integer multiplications, our construction involves squaring integers. Since squaring a large integer requires fewer basic word multiplications than multiplying two large integers, we get a speed-up. In certain architectures, multiplication takes significantly more time than other basic arithmetic operations, so we can get good savings with this approach. Our second idea is to optimize the implementation of this hash function by ignoring certain parts of the computation; moreover, we formally prove that, despite ignoring these parts of the computation, the bound ϵ on the resulting optimized hash function is still low enough to provide for a very secure MAC. One can think of this as "theoretical hacking." Specifically, the second new idea in this paper is to completely ignore some of the carry bits when performing the computation of the hash function in our basic construction. Since carry bits can be cumbersome to deal with, we can save computational effort in this manner. We stress that this savings will primarily occur when our tag size is several words long since some architectures allow you to multiply two words, and get a two-word result with all the carries "for free." At first it seems counterintuitive that we can simply ignore what appears to be a crucial part of the computation. However, we are able to obtain a bound on the resulting value of ϵ and we show that our MAC algorithms are still secure for natural choices of the parameters.

Square Hash builds on some of the ideas in the MMH construction of Halevi and Krawczyk [12]; Knudsen independently proposed a similar construction for use in block cipher design [15]. We start with an underlying hash function which is similar to the one used in MMH; however, our new hash function performs fewer multiplications. In MMH, the final carry bit of the output is ignored – in Square Hash we extend this idea by showing that we can ignore almost all of the carry bits and can still get quite a reasonable trade-off in security. Hence, in theory, Square Hash should be a strong alternative to MMH. We have implementation results on an ARM processor to substantiate this claim. Moreover, since word blocks in the input can be worked on independently, our constructions are parallelizable. We also show how to efficiently convert any Δ -universal hash function into a strongly universal hash function. Thus Square Hash has other applications besides those related to message authentication.

PREVIOUS WORK. Unconditionally secure message authentication was first studied in [9] and later in [28]. The universal hash function approach for MACs was first studied in [28] and the topic has been heavily addressed in the literature [27], [16], [25], [2], [13], [14], [26], [12]. The MMH scheme [12] is our point of departure. MMH achieves impressive software speeds and is substantially faster than many current software implementations of message authentication techniques and software implementations of universal hashing. Unfortunately, it is impossible to do precise comparisons because the available data represents simulations done on various platforms. The reader can refer to [26], [16], [5], [20]

for implementation results of various MAC schemes. This paper aims to extend the ideas in the MMH construction by exhibiting, what seems to be, a faster to compute underlying hash function and by developing some new ideas which noticeably optimize the implementation with reasonably small security costs. In addition to their message authentication applications, universal hash functions are used in numerous other settings [19], [22], [7], [28].

ORGANIZATION OF THIS PAPER. In section 2 we review the basic definitions and properties of universal hash function families and their variants. In section 3 we give the basic construction of the Square Hash, prove some basic properties, and explain why it should perform better than one of the most well-known universal hash function families. In section 4 we compare Square Hash with the MMH family of hash functions [12]. We show that all of their clever optimization techniques apply to Square Hash. We examine a novel optimization technique of ignoring all the carry bits in certain parts of the computation, and prove that the resulting construction still yields a strong message authentication scheme. Finally, in the last two sections we discuss some relevant implementation issues and give implementation results on an ARM processor.

2 Preliminaries

Let H be a family of functions going from a domain D to a range R. Let ϵ be a constant such that $1/|R| \leq \epsilon \leq 1$. The probabilities denoted below are all taken over the choice of $h \in H$.

Definition 1. *H* is a universal family of hash functions if for all $x, y \in D$ with $x \neq y$, $\Pr_{h \in H}[h(x) = h(y)] = 1/|R|$. *H* is an ϵ -almost-universal family of hash functions if $\Pr_{h \in H}[h(x) = h(y)] \leq \epsilon$.

Definition 2. Let R be an Abelian Group and let '-' denote the subtraction operation with respect to this group. Then H is a Δ -universal-family of hash functions if for all $x, y \in D$ with $x \neq y$, and all $a \in R$, $\Pr_{h \in H}[h(x) - h(y) = a] \leq 1/|R|$. H is ϵ -almost- Δ -universal if $\Pr_{h \in H}[h(x) - h(y) = a] \leq \epsilon$.

Definition 3. H is a strongly universal family of hash functions if for all $x, y \in D$ with $x \neq y$ and all $a, b \in R$, $\Pr_{h \in H}[h(x) = a, h(y) = b] \leq 1/|R|^2$. H is ϵ -almost-strongly-universal family of hash functions if $\Pr_{h \in H}[h(x) = a, h(y) = b] \leq \epsilon/|R|$.

3 Square Hash

We now describe some basic constructions of universal hash function families based on squaring. We also examine modifications that enable faster implementations at negligible costs in collision probability. In the definitions and theorems that follow, we work over the integers modulo p where p is a prime.

3.1 The Basic Construction

Definition 4. Define the SQH family of functions from Z_p to Z_p as: $SQH \equiv \{h_x : Z_p \longrightarrow Z_p | x \in Z_p\}$ where the functions h_x are defined as:

$$h_x(m) \equiv (m+x)^2 \bmod p. \tag{1}$$

Theorem 1. The family SQH is Δ -universal.

Proof. For all $m \neq n \in Z_p$, and $\delta \in Z_p$: $\Pr_x[h_x(m) - h_x(n) = \delta] = \Pr_x[(m + x)^2 - (n + x)^2 = \delta] = \Pr_x[m^2 - n^2 + 2(m - n)x = \delta] = 1/p$. The last inequality follows since for all $m \neq n \in Z_p$ and $\delta \in Z_p$ there is a unique x which satisfies the equation $m^2 - n^2 + 2(m - n)x = \delta$.

3.2 Converting From Delta-Universal to Strongly Universal

We now show how to convert any Δ -universal family of hash functions to a strongly universal family of hash functions.

Definition 5. Define the SQHU family of functions from Z_p to Z_p as: SQHU $\equiv \{h_{x,b}: Z_p \longrightarrow Z_p | x, b \in Z_p\}$ where the functions $h_{x,b}$ are defined as:

$$h_{x,b}(m) \equiv ((m+x)^2 + b) \bmod p. \tag{2}$$

Theorem 2. The family SQHU is a strongly universal family of hash functions. Proof. Follows as a corollary of the next lemma.

Lemma 1. Let $H = \{h_x : D \longrightarrow R | x \in K\}$, where R is an Abelian group and K is the set of keys, be a Δ -universal family of hash functions. Then $H' = \{h'_{x,b} : D \longrightarrow R | x \in K, \ b \in R\}$ defined by $h'_{x,b}(m) \equiv (h_x(m) + b)$ (where the addition is the operation under the group R) is a strongly universal family of hash functions.

Proof. For all $m \neq n \in D$ and all $\alpha, \beta \in R$: $\Pr_{x,b}[h'_{x,b}(m) = \alpha, h'_{x,b}(n) = \beta] = \Pr_{x,b}[h_x(m) + b = \alpha, h_x(n) + b = \beta] = \Pr_{x,b}[h_x(m) - h_x(n) = \alpha - \beta, b = \alpha - h_x(m)] = \Pr_{x,b}[h_x(m) - h_x(n) = \alpha - \beta \mid b = \alpha - h_x(m)] \cdot \Pr_{x,b}[b = \alpha - h_x(m)] = 1/|R|^2$. The last equation follows since h_x is a Δ -universal hash function and $h_x(m) - h_x(n)$ takes any value in R with equal probability.

3.3 Comparison with Linear Congruential Hash

We compare our Square Hash to the Linear Congruential Hash, which is one of the most heavily referenced Universal Hash Functions in the literature. We define the Linear Congruential Hash (LCH) family of functions to be: $LCH \equiv \{h_{x,b}: Z_p \longrightarrow Z_p | x, b \in Z_p\}$ where each of the functions $h_{x,b}$ are defined as:

$$h_{x,b}(m) \equiv mx + b \bmod p. \tag{3}$$

In most cases, the SQHU family requires less computation time than LCH. The speedup occurs because squaring an n-bit number requires roughly half the number of basic word multiplications than multiplying two n-bit numbers [18]; thus we can save when dealing with quantities that are several words long. We now compare Square Hash with the MMH construction [12].

4 Comparison with MMH

Recently Halevi and Krawczyk [12] studied a family of Δ -universal hash functions entitled MMH. MMH was originally defined by Carter and Wegman. Halevi and Krawczyk discovered techniques to speed up the software implementation at negligible costs in the collision probabilities. These hash functions are suitable for very fast software implementation. They apply to hashing variable sized data and to fast cryptographic message authentication. In this section we compare our SQH family to MMH. We show that in theory SQH is more efficient with respect to both computation time and key sizes than MMH. We also show that all of the clever software optimizations discovered by Halevi and Krawczyk for MMH can be applied to SQH as well. Finally, we further optimize Square Hash by disregarding many of the carry bits in the computation. We now describe MMH^* which is the basic non-optimized version of MMH.

4.1 Description of MMH^*

Definition 6. [12] Let k > 0 be an integer. Let $x = \langle x_1, \ldots, x_k \rangle$, and $m = \langle m_1, \ldots, m_k \rangle$, $x_i, m_i \in Z_p$, The MMH* family of functions from Z_p^k to Z_p is defined as follows: MMH* $\equiv \{g_x : Z_p^k \longrightarrow Z_p \mid x \in Z_p^k\}$ where the functions g_x are defined as

$$g_x(m) = m \cdot x = \sum_{i=1}^k m_i x_i \bmod p \tag{4}$$

Theorem 3. [Halevi and Krawczyk]: MMH^* is a Δ -universal family of hash functions.

Halevi and Krawczyk [12] also discussed a way to generalize these functions so that their range is Z_p^l rather than just Z_p . This can be done via a Cartesian product type idea due to Stinson [27]. Specifically, we hash the message l times using l independently chosen keys and we concatenate the hashes. This yields a collision probability of $1/p^l$. At first this requires a much larger key size, but that can be reduced by applying a Toeplitz matrix type idea due to Krawczyk [16]; namely (for the case l = 2), choose k + 1 scalars x_1, \ldots, x_{k+1} and set the first key to be $\langle x_1, \ldots, x_k \rangle$ and the second key to be $\langle x_2, \ldots, x_{k+1} \rangle$. The collision probability reduces to $1/p^2$.

4.2 A Variant of Square Hash Similar to MMH^*

Definition 7. Let k > 0 be an integer. Let $x = \langle x_1, \ldots, x_k \rangle$, and $m = \langle m_1, \ldots, m_k \rangle$, $x_i, m_i \in Z_p$. The SQH^* family of functions from Z_p^k to Z_p is defined as follows: $SQH^* \equiv \{g_x : Z_p^k \longrightarrow Z_p \mid x \in Z_p^k\}$ where the functions g_x are defined as

$$g_x(m) = \sum_{i=1}^k (m_i + x_i)^2 \bmod p$$
 (5)

Theorem 4. SQH^* is a Δ -universal family of hash functions.

Proof. Let $m \neq n \in Z_p^k$ with $m = \langle m_1, \ldots, m_k \rangle$, $n = \langle n_1, \ldots, n_k \rangle$ and $m_i, n_i \in Z_p$. Let $\delta \in Z_p$. Since $m \neq n$ there is some i for which $m_i \neq n_i$. WLOG, suppose $m_1 \neq n_1$. Now, we show: $\forall x_2, \ldots, x_k \Pr_{x_1}[g_x(m) - g_x(n) = \delta] = 1/p$ (where $x = \langle x_1, \ldots, x_k \rangle$) which implies the lemma. So, $\Pr_{x_1}[g_x(m) - g_x(n) = \delta] = \Pr[\sum_{i=1}^k (x_i + m_i)^2 - \sum_{i=1}^k (x_i + n_i)^2 = \delta] = \Pr[2(m_1 - n_1)x_1 = \delta - m_1^2 + n_1^2 - \sum_{i=2}^k (x_i + m_i)^2 + \sum_{i=2}^k (x_i + n_i)^2] = 1/p$. The last equation follows since $(m_1 - n_1) \neq 0$ implies that there is a unique $x_1 \in Z_p$ satisfying the equation inside the probability.

4.3 Comparing SQH^* to MMH^*

 SQH^* should be faster than MMH^* because squaring can be implemented so it requires roughly half the number of basic word multiplications as multiplying two numbers [18]. Since multiplication is relatively expensive on many architectures, we may save considerably. Halevi and Krawczyk made several clever software optimizations on MMH; the same optimizations apply to SQH as well.

4.4 Speeding up MMH^*

Here is the definition of MMH_{32} , an optimized version of MMH^* , which appeared in [12]:

Definition 8. Set $p = 2^{32} + 15$ and k = 32. Let $x = \langle x_1, ..., x_k \rangle$, and $m = \langle m_1, ..., m_k \rangle$, $x_i, m_i \in Z_p$. Define the MMH₃₂ family of functions from $(\{0, 1\}^{32})^k$ to $\{0, 1\}^{32}$ as: MMH₃₂ $\equiv \{h_x : (\{0, 1\}^{32})^k \longrightarrow \{0, 1\}^{32} \mid x \in (\{0, 1\}^{32})^k$ where the functions h_x are defined as

$$h_x(m) = (((\sum_{i=1}^k m_i x_i) \bmod 2^{64}) \bmod (2^{32} + 15)) \bmod 2^{32}$$
 (6)

Theorem 5. [Halevi and Krawczyk]: MMH_{32} is an ϵ -Almost- Δ -Universal family of hash functions with $\epsilon \leq 6 \cdot 2^{-32}$.

The same optimization applies to SQH^* .

4.5 Speeding up SQH^*

Here is a variant of Square Hash, called SQH_{asm} , which is suited for assembly language implementation.

Definition 9. Let l and k be positive integers, and let $2^l . Let <math>x = \langle x_1, \ldots, x_k \rangle$, and $m = \langle m_1, \ldots, m_k \rangle$, $x_i, m_i \in Z_p$. The SQH_{asm} family of functions from Z_p^k to $\{0,1\}^l$ is defined as follows: $SQH_{asm} \equiv \{g_x : Z_p^k \longrightarrow \{0,1\}^l \mid x \in Z_p^k\}$ where the functions g_x are defined as

$$g_x(m) = \left(\left(\sum_{i=1}^k (m_i + x_i)^2 \right) \bmod p \right) \bmod 2^l$$
 (7)

Theorem 6. SQH_{asm} is an ϵ -almost- Δ -universal family of hash functions with $\epsilon \leq 3 \cdot 2^{-l}$.

Proof. Let $\delta \in \{0,1\}^l$ be chosen arbitrarily. Let $m \neq n$ be arbitrary message vectors. Let x be the key such that $h_x(m) - h_x(n) \equiv \delta \pmod{2^l}$, where $h \in SQH^*$. Equivalently, $h_x'(m) - h_x'(n) \equiv \delta \pmod{2^l}$ where $h' \in SQH_{asm}$. Now, both $h_x(m)$ and $h_x(n)$ are in the range $0, \ldots, p-1$. Therefore, their difference taken over the integers lies in the range $-p+1, \ldots, p-1$. If we denote $p=2^l+t$ where $0 < t < 2^{l-1}$ then:

$$h_x'(m) - h_x'(n) \in \begin{cases} \{\delta, \delta - 2^l\} & t \le \delta \le 2^l - t \\ \{\delta - 2^l, \delta, \delta + 2^l\} & 0 \le \delta \le t - 1 \\ \{\delta, \delta - 2^l, \delta - 2^{l+1}\} & 2^l - t < \delta \le 2^l - 1 \end{cases}$$

Therefore, there are at most three values for the quantity $h'_x(m) - h'_x(n)$ which cause $h_x(m) - h_x(n) \equiv v \pmod{2^l}$. Since SQH^* is a Δ -universal hash function, it follows that for any $\delta' \in \{0,1\}^l$ there is at most one choice of the key x for which $h'_x(m) - h'_x(n) \equiv \delta' \mod p$. Therefore, at most 3 keys satisfy the equation $h_x(m) - h_x(n) \equiv \delta \pmod{2^l}$. So, $Pr_x[h_x(m) - h_x(n) \equiv \delta \pmod{2^l}] \leq 3 \cdot 2^{-l}$. \square

4.6 A Further Speed-Up

There is a minor weakness in SQH_{asm} . The values m_i and x_i may each be l+1 bits long. We would, however, like to make l the word size of the machine on which we are implementing our code (typically l=32 or 64) in order to speed up computations. Having to deal with l+1 bit quantities means that we have to store and square several extra words. A first solution is to restrict both m_i and x_i to be at most l bits. Unfortunately, $m_i + x_i$ may be an l+1 bit quantity which means we still need to store and square extra words. It turns out that we simply can ignore the most significant bit of $m_i + x_i$ at just a minor cost in the important statistical properties of the new hash function. We give another Square Hash variant and prove that it performs well.

Definition 10. Let l and k be positive integers with $2^l . Let <math>x = \langle x_1, \ldots, x_k \rangle$, and let $m = \langle m_1, \ldots, m_k \rangle$, $x_i, m_i \in \{0, 1\}^l$. Define SQH_{asm2} family of functions from $(\{0, 1\}^l)^k$ to $\{0, 1\}^l$ as: $SQH_{asm2} \equiv \{g_x : (\{0, 1\}^l)^k \longrightarrow \{0, 1\}^l \mid x \in \{0, 1\}^l\}$ where the functions g_x are defined as

$$g_x(m) = \left(\left(\sum_{i=1}^k ((m_i + x_i) \bmod 2^l)^2 \right) \bmod p \right)$$
 (8)

So, all we are doing is ignoring the most significant bit of $x_i + m_i$. This means that the sum will fit into l bits, which means that we do not have to use an extra word to both store and square.

Theorem 7. SQH_{asm2} is an ϵ -almost- Δ -universal family of hash functions with $\epsilon \leq 2 \cdot 2^{-l}$.

Proof. Let $m \neq n \in \{0,1\}^l$ with $m = \langle m_1, \ldots, m_k \rangle$, $n = \langle n_1, \ldots, n_k \rangle$ and $m_i, n_i \in \{0,1\}^l$. Let $\delta \in Z_p$. Since $m \neq n$ there is some i for which $m_i \neq n_i$. WLOG, suppose $m_1 \neq n_1$. Now, we show for all $x_2, \ldots, x_k \Pr_{x_1}[g_x(m) - g_x(n) = \delta] \leq 2/2^l$ (where $x = \langle x_1, \ldots, x_k \rangle$) which implies the lemma. First, let

$$A = \sum_{i=2}^{k} ((x_i + m_i) \bmod 2^l)^2 \bmod p, \text{ and let } B = \sum_{i=2}^{k} ((x_i + n_i) \bmod 2^l)^2 \bmod p.$$

Then, $\Pr_{x_1}[g_x(m)-g_x(n)=\delta]=\Pr_{x_1}[(((x_1+m_1) \bmod 2^l)^2+A)-(((x_1+n_1) \bmod 2^l)^2+B)\equiv \delta\pmod p]=\Pr_{x_1}[((x_1+m_1) \bmod 2^l)^2-((x_1+n_1) \bmod 2^l)^2\equiv B-A+\delta\pmod p].$ Since x_1 and m_1 are both l bit quantities, their sum will be at most $2^{l+1}-2$, which means that to reduce this quantity mod 2^l we have to subtract off at most 2^l . Therefore,

$$((x_1 + m_1) \bmod 2^l) = x_1 + m_1 - 2^l c(x_1, m_1), \tag{9}$$

where $c(x_1, m_1)$ is some value in $\{0, 1\}$. In this case, c is the carry bit associated with adding x_1 and m_1 . Similarly, we can write $((x_1 + n_1) \mod 2^l) = x_1 + n_1 - 2^l c(x_1, n_1)$. Replacing these equations into the above and performing some arithmetic manipulation, we get:

$$\Pr_{x_1}[((x_1 + m_1) \bmod 2^l)^2 - ((x_1 + n_1) \bmod 2^l)^2 \equiv B - A + \delta \pmod{p}]
= \Pr_{x_1}[2x_1((m_1 - n_1) + (c(x_1, n_1) - c(x_1, m_1))2^l) \equiv \delta' \pmod{p}].$$

Where

$$\delta' = B - A + \delta + (n_1 - c(x_1, n_1)2^l)^2 - (m_1 - c(x_1, m_1)2^l)^2.$$
 (10)

Now, $(c(x_1, n_1) - c(x_1, m_1))2^l \in \{-2^l, 0, 2^l\}$. However, since $m_1, n_1 \in \{0, 1\}^l$ and since $m_1 \neq n_1$, it follows that $(m_1 - n_1) \in \{(1 - 2^l), \ldots, -1, 1, \ldots, (2^l - 1)\}$ which implies that $((m_1 - n_1) + (c(x_1, n_1) - c(x_1, m_1))2^l) \neq 0$, and for a given $c(x_1, m_1)$ and $c(x_1, n_1)$ there is at most one value of x_1 satisfying the above equations. Finally, we have

$$\Pr_{x_1}[2x_1((m_1 - n_1) + (c(x_1, n_1) - c(x_1, m_1))2^l) \equiv \delta']$$

$$\leq \Pr_{x_1}[2x_1((m_1 - n_1) - 2^l) \equiv \delta' \pmod{p}|c(x_1, n_1) - c(x_1, m_1) = -1]$$

$$+ \Pr_{x_1}[2x_1(m_1 - n_1) \equiv \delta' \pmod{p}|c(x_1, n_1) - c(x_1, m_1) = 0]$$

$$+ \Pr_{x_1}[2x_1((m_1 - n_1) + 2^l) \equiv \delta' \pmod{p}|c(x_1, n_1) - c(x_1, m_1) = 1] \leq 3/2^l.$$

This gives us a bound of $3/2^l$. We can improve this to $2/2^l$ by observing that for fixed values of m and n, $c(x_1, n_1) - c(x_1, m_1)$ cannot simultaneously take on the values +1 and -1 for varying choices of x_1 . In particular, if $n_1 > m_1$ then we claim that $c(x_1, n_1) - c(x_1, m_1) \ge 0$. This follows because $c(x_1, n_1) - c(x_1, m_1) = -1$ implies $x_1 + n_1 < 2^l$ and $x_1 + m_1 \ge 2^l$ which implies that $m_1 > n_1$. Similarly, it can be shown that $n_1 < m_1$ implies $c(x_1, n_1) - c(x_1, m_1) \le 0$. Thus $c(x_1, n_1) - c(x_1, m_1)$ takes on at most two possible values and ϵ is bounded by $2/2^l$. \square

4.7 Ignoring Carry Bits in the Computation

We now describe a way to further speed up Square Hash at a small tradeoff in the collision probability. The idea is novel, and can be applied not only to MMH but perhaps to other constructions of universal hash functions. Basically, we show that we can ignore many of the carry bits in the computation of Square Hash and still get very strong performance for cryptographic applications. In some sense this extends the ideas of Halevi and Krawczyk who sped up MMH by ignoring only the most significant carry bit. We start by describing the notion of carry bits and explain why computation can speed up if you ignore them. We then define variants of Square Hash in which you can ignore some of the carry bits, and show that the resulting performance is still excellent for cryptographic applications. Finally, we define yet another variant of Square Hash in which you can ignore even more carry bits and show that the performance is still strong for cryptographic applications under suitable settings for the parameters.

Carry Bits When two words are added, there is usually an overflow or carry that takes place in the computation. For example, if the word size is 8, and you compute 11001001 + 10010001 you get 101011010. Since the word size is 8, the most significant bit 1 is called the carry or overflow bit because it overflowed from the usual 8 bit word size. Now, when arithmetic operations are performed on very long integers, as is usually the case for cryptographic applications, the carry bits between individual word operations are used for the next operation. So, if the word size is 8, and you are trying to compute 1011010100110101 + 1010101111100101 then the computation is broken up into parts. First, each bit string is broken up to take word size into account. The first string is broken up into two parts which we label A and B respectively: A = 10110101 and B =00110101. The second string would be broken up into two parts: C = 10101011and D = 11100101. Now, the computation is carried out as follows: first we compute B + D store the answer in a word, and store the carry c_0 separately. Denote by E the word in which we store B + D. Then E = 00011010 and the carry bit $c_0 = 1$. Now, we compute $F = A + C + c_0$ and store the carry bit in c_1 . Then F = 01100001 and the carry bit c_1 is 1. The total answer is the concatenation c_1FE : 10110000100011010. So, it is necessary to keep track of a carry bit as you do the computations on integers that require more than one word to represent. Unfortunately, certain instructions on processors do not deal with carry bits effectively (for example the Multiply with Accumulate instruction on the ARM). Also, even if an instruction saves the carry bit, this information may get destroyed when other instructions are executed. In addition, most high level programming languages do not deal with carry bits effectively; this increases the computation time of arithmetic instructions over integers that are several words long because it becomes necessary to explicitly keep track of the carry bit. High level programming languages are, however, preferable because they are portable and they facilitate the task of programming. We show that we can overcome these dilemmas by ignoring the carry bits altogether. We call these variants of Square Hash SQH_c and SQH_{c2} since they can be effectively implemented with

high level programming languages such as C. We can prove that we get strong performance despite ignoring what seems to be a crucial part of the computation.

Ignoring Carry Bits in the Outer Summation We describe a preliminary speedup in which we ignore the carry bits in the outer summation, and show that we still get a powerful approximation to a Δ -universal hash. Let us denote by $\mathcal{C}(\Sigma_{i=1}^n a_i)$ the value you get if you compute the sum $\Sigma_{i=1}^n a_i$ but ignore the carry bits between the words. For example, if you let the word size be 8 and compute $a_1 = 101101010110110101$ and $a_2 = 10101011111100101$ as in the above example, then $\mathcal{C}(\Sigma_{i=1}^2 a_i) = 011000000011010$. The normal sum $a_1 + a_2$ is 10110000100011010. But recall that the 9th least significant bit is the result of the carry from the summation of first pair of words, and the most significant bit is the result of the carry from the second pair of words. Since you are ignoring the carry bits, the function $\mathcal{C}(\Sigma_{i=1}^n a_i)$ can be implemented much more efficiently than just the normal sum $\sum_{i=1}^{n} a_i$. This is especially true if the a_i are large integers and each require several words in order to store. We now formally define a new variant of Square Hash and show that it still gives us strong performance.

Definition 11. Let l and k be positive integers with $2^{l} . Let$ $x = \langle x_1, \ldots, x_k \rangle$, and $m = \langle m_1, \ldots, m_k \rangle$, $x_i, m_i \in Z_p$. The SQH_c family of functions from Z_p^k to Z_p is defined as follows: $SQH_c \equiv \{g_x : Z_p^k \longrightarrow Z_p \mid x \in Z_p^k\}$ where the functions g_x are defined as

$$g_x(m) = (\mathcal{C}(\sum_{i=1}^k (m_i + x_i)^2) \bmod p)$$
 (11)

Theorem 8. Let l be the word size of the architecture on which you are computing and let w be the number of words it takes to store x_i . Then SQH_c is an ϵ -almost- Δ -universal family of hash functions with $\epsilon \leq 3^{2w}/2^{lw}$.

Proof. Fix a value $a \in \mathbb{Z}_p$ and let $m = \langle m_1, \ldots, m_k \rangle \neq m' = \langle m'_1, \ldots, m'_k \rangle$ be your two messages. Assume wlog that $m_1 \neq m'_1$. We prove that for any choice of $(x_2, \ldots, x_k \Pr_{x_1} [g_x(m) - g_x(m') = a \mod p] \le 3^{2w} / 2^{lw} \text{ (where } x = \langle x_1, \ldots, x_k \rangle)$ which implies the theorem. Now, let us fix some choice of x_2, \ldots, x_k and let $s = \mathcal{C}(\sum_{i=2}^{k} (m_i + x_i)^2)$. Then,

$$C(\sum_{i=1}^{k} (m_i + x_i)^2) = (x_1 + m_1)^2 + s - c$$
(12)

where $c \in \{0,1\}^{2l+1}$ is a "correction vector" in which the *ith* bit of c (counting from the left) contains a 1 if there was an overflow of 1 at that position (and contains a 0 otherwise). In the example above with a_1 and a_2 the correction vector c is: 1000000100000000. Similarly, if we let $s' = C(\sum_{i=2}^k (m'_i + x_i)^2)$ then

$$C(\sum_{i=1}^{k} (m_i' + x_i)^2) = (x_1 + m_1')^2 + s' - c'$$
(13)

where c' is the associated correction vector. Therefore,

$$\Pr_{x_1}[g_x(m) - g_x(m') \equiv a \pmod{p}]$$

$$= \Pr_{x_1}[(x_1 + m_1)^2 + s - c - (x_1 + m_1')^2 - s' + c' \equiv a \pmod{p}]$$

$$= \Pr_{x_1}[x_1 \equiv (a + (c - c') + s' - s + m_1'^2 - m_1^2)/2(m_1 - m_1') \pmod{p}]$$

$$\leq (\text{The number of distinct values } c - c' \text{ can take}) \cdot 2^{-lw}.$$

So we must derive a bound for the number of distinct values c-c' can take. Now, c and c' consist mostly of 0's. In fact, the only positions of c in which there could be a 1 are the ones where there could be a carry – and since carries only occur at the boundaries between words, only bits $l+1, 2l+1, 3l+1, \ldots, 2wl+1$ can possibly contain a 1. The same hold true for c'. In the il+1 bit $c_{il+1}-c'_{il+1} \in \{-1,0,1\}$ for $1 \le i \le 2l$. Since there are only 2w bits that can get affected and 3 different values for their difference, the total number of different vectors c-c' is bounded by 3^{2w} . So, we have that $\Pr_{x_1}[g_x(m)-g_x(m')\equiv a(\text{mod}p)] \le 3^{2w}/2^{lw}$ – which proves the theorem.

Now, observe that the quantity $3^{2w}/2^{lw}$ is actually rather small. We see this if we substitute suitable values for the parameters. If the word size l is 32 bits, then a computationally unbounded adversary can forge a MAC tag of size 2,3,4, or 5 words with probability at most 2^{-57} , 2^{-86} , 2^{-115} , and 2^{-144} respectively. These are negligible and are smaller that what one may need for a reasonably secure MAC. This leads to the question of whether we can optimize further at a slightly greater cost in security. The next section works towards this aim by showing that we can ignore even more carry bits at an increased cost in collision probability.

Ignoring Carry Bits When Squaring Since the process of squaring can be expressed entirely in terms of doing basic word multiplications, shifts, and add operations, we can consider the idea of ignoring the carry bits when performing a squaring operation to further speed up our hash functions. We show that if we also ignore the carry bits that occur when the quantity $(x_i + m_i)$ is squared, then the resulting function still yields a close approximation to a δ -universal hash for suitable values for the parameters. So let's denote by $C_2(a_i^2)$ the quantity obtained if you ignore the carry bits in the computation of a_i^2 .

Definition 12. Let l and k be positive integers, with $2^l . Let <math>x = \langle x_1, \ldots, x_k \rangle$, and $m = \langle m_1, \ldots, m_k \rangle$, $x_i, m_i \in \{0, 1\}^l$. The SQH_{c2} family of functions from $(\{0, 1\}^l)^k$ to Z_p is defined as follows: $SQH_{c2} \equiv \{g_x : (\{0, 1\}^l)^k \longrightarrow Z_p \mid x \in (\{0, 1\}^l)^k\}$ where the functions g_x are defined as

$$g_x(m) = (\mathcal{C}(\sum_{i=1}^k \mathcal{C}_2((m_i + x_i)^2)) \bmod p).$$
 (14)

Note that we ignore carry bits when we square and when we take the sum over the $(x_i + m_i)^2$. However, we do not ignore the carry bits when we actually compute $(x_i + m_i)$. It is possible that we may be able to get away with ignoring these carry bits as well. We now state our main theorem about how well this new family of hash functions performs:

Theorem 9. Let l be the word size of the architecture on which you are computing and let w be the number of words it takes to store x_i . Then SQH_{c2} is an ϵ -almost- Δ -universal family of hash functions with $\epsilon \leq (\prod_{i=1}^{w} (4i+1)^2)/2^{lw}$.

Proof. (sketch) The proof uses similar ideas to the proof of the previous theorem about ignoring carry bits in the outer summation. In particular, we define correction vectors c and c' in the same manner as before, and bound the number of distinct possibilities for their difference c-c'. We first observe that for ith word of c (counting from the right for $1 \le i \le w$) there are 1's in at most the $\log(2i+1)$ least significant bit positions of that word – the remaining bits must be 0. So, only the least significant $\log(2i+1)$ bits in word i are undetermined. Similarly, for word j with $w+1 \le j \le 2w$, the least significant $\log(4w+3-2j)$ are undetermined. Moreover, we can show that if the b least significant bits of each of two different words are undetermined, then the difference of those two words can take on at most $2^{b+1}-1$ different values. The number of distinct possible values for c-c' is the product of the number of different possible values each of the individual words can take. This equals: $\prod_{i=1}^w (2^{\log(2i+1)+1}-1) \cdot \prod_{j=m+1}^{2w} (2^{\log(4w+3-2j)+1}-1) = (\prod_{i=1}^w 2^{\log(2i+1)+1}-1)^2 = (\prod_{i=1}^w 4i+1)^2$.

Although this expression looks large, for suitable values of the parameters it still gives good security. Keep in mind that typically $1 \le w \le 5$. In particular, if the word size l is 32 bits, and we hash down to 2,3,4, or 5 words, then *computationally unbounded* adversaries will fail to forge the MAC with probability better than 2^{-53} , 2^{-77} , 2^{-101} , or 2^{-124} respectively.

4.8 Fully Optimized Square Hash

We present the fully optimized version of Square Hash:

Definition 13. Let l and k be positive integers with $2^l . Let <math>x = \langle x_1, \ldots, x_k \rangle$, and $m = \langle m_1, \ldots, m_k \rangle$, $x_i, m_i \in \{0, 1\}^l$. The SQH_E family of functions from $(\{0, 1\}^l)^k$ to $\{0, 1\}^l$ is defined as follows: $SQH_E \equiv \{g_x : (\{0, 1\}^l)^k \longrightarrow \{0, 1\}^l \mid x \in \{0, 1\}^l\}$ where the functions g_x are defined as

$$g_x(m) = (\mathcal{C}(\sum_{i=1}^k \mathcal{C}_2(((m_i + x_i) \bmod 2^l)^2)) \bmod p) \bmod 2^l$$
 (15)

Theorem 10. Let l be the word size on which you are computing and w is the total number of words needed to store x_i . Then SQH_E is an ϵ -almost- Δ -universal family of hash functions with $\epsilon \leq (6 \cdot \prod_{i=1}^{w} (4i+1)^2)/2^{lw}$

Proof. Combine the proofs and statements of previous theorems.

4.9 Comparison to NMH

At the end of their paper, Halevi and Krawczyk [12] briefly discussed another family of Δ -universal hash functions called NMH. It would be interesting to do a detailed comparison between NMH and SQH that studies speed and required key sizes. Another interesting area for future research would be to apply some of our techniques of ignoring carry bits to MMH and NMH.

5 Considerations on Implementing Square Hash

In this section, we discuss various important implementation considerations, and in the next we give actual implementation results. To start with, Square Hash should be faster since we use squaring instead of multiplication. The speed-up factor for squaring an n word integer versus multiplying two n word integers is (n-1)/2n. Typically, MACs have tag sizes between 32 and 160 bits, depending on the level of security needed. Therefore, on 32-bit architectures, $1 \le n \le 5$ and we get speed up factors of %0, %25, %33, %38, and %40 for the different values of n. Now, on most slower architectures, multiplications require many more clock cycles than other simple arithmetic operations such as addition. For example, on the original Pentium processor, the ratio between number of clock cycles for unsigned multiplication versus addition is about 5:1. This ratio probably gets much larger on weaker processors such as those on cellular phones, embedded devices, smart-cards, etc., Moreover, for these types of smaller processors, word sizes may be smaller, hence the number of words we multiply increases, and the savings we achieve by using squaring rather than multiplication greatly increases. Thus, we recommend using Square Hash on such architectures. On some of the more modern processors such as the Pentium Pro and Pentium II, multiplications do not take much more time than additions (closer to 2:1, [8]), so Square Hash is not advantageous is such cases.

Another important implementation consideration is the memory architecture of the processor on which you are implementing. In our case, we need extra data registers to quickly implement squaring. On Pentium architectures there are only 4 32-bit data registers [8]. Hence, we may need to make additional memory references which could slow things down. On the other hand, the PowerPC has 32 32-bit general purpose registers [1], which allows us to get fast squaring.

6 Implementation Results

We used the ARM (i.e. ARM7) processor to create hand optimized assembly code to compare speeds of various algorithms. The ARM7 is a popular RISC processor and is used inside cellular phones, PDAs, smartcards, etc. It is a 32 bit processor with 16 general purpose registers. Basic operations like addition require 1 cycle whereas multiplication usually requires 6 cycles.

Our results show a significant speedup for square hash over MMH, and thus validate our theoretical results. For long messages and same or better security than MMH, square hash is 1.31 times faster than MMH (Table 1).

	MMH	SQH1	SQH2	HMAC-SHA1
			(some carries dropped)	
Cycles	2061	1659	1575	4000+
Speedup over MMH			1.31x	.52x
Speedup over SHA1			2.54x	1x
$Security - \epsilon$	6.25 290	6 296	2.53 290	
Code Size (bytes)	408	3040	2704	4000+
Hash key Size	2208	2112	2112	
(random bits)	(2112+96)			

Table 1. Assembly implementations on ARM7 with 96 bit output and input block size of 2112 bits.

Message authentication using universal hash functions is performed by breaking up a long message (e.g 1Mbyte) in to smaller blocks (e.g. 2112 bits) and reducing each block, using an equivalent size hash key, down to the size of the MAC output or tag size (e.g. 96 bits). This is repeated via a tree hash until the final tag is output. Tree hash adds about 10% overhead to both square hash and MMH [12] and we omit it in the calculations presented in the tables for purposes of simplifying comparison. The security parameter ϵ as reported in the tables would have to be multiplied by the height of the tree [12].

We report results for a tag size of 96 bits since we believe it is a popular choice for message authentication in Internet standards (e.g. HMAC). Larger output sizes of 128 and 160 bits could further improve speedup factors due to greater savings on multiplications. We also report cycle counts for SHA1 on an ARM7 to verify that we are faster than traditional non-universal hash based MACs (e.g. HMAC). To create the MAC, in actual use, MMH and square hash would have to encrypt the 96 bit output and HMAC-SHA1 would need to perform a further SHA1. We exclude this in the cycle counts in the tables to simplify comparison.

First for 2112-bit blocks (a multiple of 96) we compare MMH, SQH1, SQH2, and HMAC-SHA1. SQH1 is the basic square hash function SQH_{asm} with the minor optimization of SQH_{asm2} giving an overall security of $\frac{6}{2^{96}}$ compared to the security of $\frac{6.25}{2^{90}}$ for 96 bit MMH. SQH2 is the same as SQH1, except that some carry bits are dropped in the squaring until the security is similar or better than that of MMH. As a result of dropping some carries, computation time decreases.

SHA1 requires more than 1000 operations on 512-bit input and thus requires more than 4000 operations on 2112 bit input. All 3 universal hashes are significantly faster than the SHA1 based MAC. SQH1 is 1.24 times as fast as MMH and SQH2 is 1.31 times as fast as MMH. Code sizes are somewhat large because of loop unrolling. Without unrolling additional computational time will be added to all three universal hashes to handle looping. The hash key (random bits) for MMH is 96 bits larger than square hash if the Toeplitz construction is used [12].

	MMH	SQH1	SQH2	HMAC-SHA1
			(some carries	
			dropped)	
Cycles	1086	856	816	2000
Speedup over MMH	1x	1.27x	1.33x	.54x
Speedup over SHA1	1.84x	2.34x	2.45x	1x
Code Size	220	1544	1384	4000+
	-		1056	
(random bits)	(1056+96)			

Table 2. Assembly implementations on ARM7 with 96 bit output and input block size of 1056 bits.

In Table 2 we also report cycle counts for 1056-bit blocks. Since 1024 bit blocks, as used by [12], are not a multiple of 96, we used 1056 (a multiple of 32 and 96) as the input length. We ran experiments with messages that had the same size as the tag, and we noticed similar speedups. We also tested C versions of MMH and square hash and we saw similar speedups.

Table 3 gives break downs of the instruction and cycle counts for both MMH and SQH2. In the future we hope to experiment with other processors.

	Instructions	Cycles
MMH 66 (2112/32) words	105	687
key + message loading	28	188
Multiply + Accumulate	66	461
mod p reduction	5	9
function call	6	29
Total (3 times MMH 66 words)	105	2061
SQH2 3 words	29	69
key + message loading	2	10
Multiply	5	30
Multiply + Accumulate	1	7
Adds	21	21
22 (2112/96) times SQH2 (3 words)	638	1518
mod p reduction	16	25
function overhead	10	32
Total	664	1575

Table 3. MMH and SQH2 cycle count break downs: 96 bit output and block size of 2112 bits.

7 Conclusion

We described a new family of universal hash functions geared towards high speed message authentication. On some platforms, our hash functions appear to be faster than the MMH family, which itself is considered to be one of the fastest universal hash function implementations. We also introduced additional techniques for speeding up our constructions. These constructions and techniques are general and may be of independent interest.

Acknowledgments

The authors would like to thank Shai Halevi, Dan Heer, Umair Patel, Dana Richardson, and Ganesh Sundaram for their help. We also thank the anonymous referees and program committee for their comments and suggestions.

References

- 1. Rhapsody Assembler Reference. Available via http://developer.apple.com/.
- 2. M. Atici and D. Stinson. Universal hashing and multiple authentication. In *Proc. CRYPTO 96*, Lecture Notes in Computer Science. Springer-Verlag, 1996.
- M. Bellare, R. Canetti, , and H. Krawczyk. Keying hash functions for message authentication. In *Proc. CRYPTO 96*, Lecture Notes in Computer Science. Springer-Verlag, 1996.
- 4. M. Bellare, J. Killian, and P. Rogaway. The security of cipher block chaining. In *Proc. CRYPTO 94*, Lecture Notes in Computer Science. Springer-Verlag, 1994.
- 5. A. Bosselaers, R. Govaerts, and J. Vandewalle. Fast hashing on the Pentium. In *Proc. CRYPTO 96*, Lecture Notes in Computer Science. Springer-Verlag, 1996.
- G. Brassard. On computationally secure authentication tags requiring short secret shared keys. In Proc. CRYPTO 82, Lecture Notes in Computer Science, pages 79–86, 1982.
- L. Carter and M. Wegman. Universal hash functions. Journal of Computer and System Sciences, 18:143–144, 1979.
- 8. S. P. Dandamudi. Introduction to Assembly Language Programming From 8086 to Pentium Processors. Springer-Verlag New York, 1998.
- E. Gilbert, F.M. Williams, and N. Sloane. Codes which detect deception. Bell System Technical Journal, 53(3):405–424, 1974.
- 10. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):210–217, 1986.
- O. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. Siam Journal of Computing, 17(2):281–308, 1988.
- 12. S. Halevi and H. Krawczyk. MMH: Message authentication in software in the gbit/second rates. In *Proceedings of the 4th Workshop on Fast Software Encryption*, Lecture Notes in Computer Science. Springer-Verlag, 1997.
- 13. T. Helleseth and T. Johansson. Universal hash functions from exponential sums over finite fields. In *Proc. CRYPTO 96*, Lecture Notes in Computer Science. Springer-Verlag, 1996.
- 14. T. Johansson. Bucket hashing with small key size. In *Proc. EUROCRYPT 97*, Lecture Notes in Computer Science. Springer-Verlag, 1997.

- 15. L. Knudsen. Truncated and higher order differentials. In *Proceedings of the* 2nd Workshop on Fast Software Encryption, Lecture Notes in Computer Science. Springer-Verlag, 1995.
- H. Krawczyk. LFSR-based hashing and authentication. In Proc. CRYPTO 94, Lecture Notes in Computer Science. Springer-Verlag, 1994.
- H. Krawczyk. New hash functions for message authentication. In Proc. EURO-CRYPT 95, Lecture Notes in Computer Science. Springer-Verlag, 1995.
- A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.
- M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-rackoff revisited. *Journal of Cryptology*, 1999. To Appear. Extended abstract in: Proc. 29th Ann. ACM Symp. on Theory of Computing, 1997, pp. 189-199.
- W. Nevelsteen and B. Preneel. Software performance of universal hash functions. In Proc. EUROCRYPT 99, Lecture Notes in Computer Science. Springer-Verlag, 1999.
- National Bureau of Standards. FIPS publication 46: Data encryption standard, 1977. Federal Information Processing Standards Publication 46.
- 22. S. Patel, Z.A. Ramzan, and G. Sundaram. Towards making Luby-Rackoff ciphers practical and optimal. In *Proceedings of Sixth Workshop on Fast Software Encryption*, March 1999. To Appear.
- B. Preneel and P.C. van Oorschot. On the security of two MAC algorithms. In Proc. EUROCRYPT 96, Lecture Notes in Computer Science. Springer-Verlag, 1996.
- 24. R. Rivest. The MD5 message digest algorithm. IETF RFC-1321, 1992.
- 25. P. Rogaway. Bucket hashing and its application to fast message authentication. In *Proc. CRYPTO 95*, Lecture Notes in Computer Science. Springer-Verlag, 1995.
- V. Shoup. On fast and provably secure message authentication based on universal hashing. In Proc. CRYPTO 96, Lecture Notes in Computer Science. Springer-Verlag, 1996.
- 27. D. Stinson. Universal hashing and authentication codes. *Designs, Codes, and Cryptography*, 4:369–380, 1994.
- M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions

Jee Hea An and Mihir Bellare

Dept. of Computer Science & Engineering, University of California at San Diego 9500 Gilman Drive, La Jolla, CA 92093, USA {jeehea,mihir}@cs.ucsd.edu

URL: http://www-cse.ucsd.edu/users/{jeehea,mihir}

Abstract. Practical MACs are typically designed by iterating applications of some fixed-input-length (FIL) primitive, namely one like a block cipher or compression function that only applies to data of a fixed length. Existing security analyses of these constructions either require a stronger security property from the FIL primitive (eg. pseudorandomness) than the unforgeability required of the final MAC, or, as in the case of HMAC, make assumptions about the iterated function itself. In this paper we consider the design of iterated MACs under the (minimal) assumption that the given FIL primitive is itself a MAC. We look at three popular transforms, namely CBC, Feistel and the Merkle-Damgård method, and ask for each whether it preserves unforgeability. We show that the answer is no in the first two cases and yes in the third. The last yields an alternative cryptographic hash function based MAC which is secure under weaker assumptions than existing ones, although at a slight increase in cost.

1 Introduction

Directly (from scratch) designed cryptographic primitives (for example block ciphers or compression functions) are typically "fixed input-length" (FIL): they operate on inputs of some small, fixed length. However, usage calls for "variable input-length" (VIL) primitives: ones that can process inputs of longer, and varying lengths. Much cryptographic effort goes into the problem of transforming FIL primitives to VIL primitives. (To mention just two popular examples: the various modes of operation of block ciphers address this problem when the given FIL primitive is a block cipher and the desired VIL primitive a data encryption scheme; and the Merkle-Damgård iteration method [16,11] addresses this problem when the given FIL primitive is a collision-resistant compression function and the desired VIL primitive is a collision-resistant hash function.) In this paper, we will address this problem for the design of VIL-MACs in the case where the given FIL primitive is itself a MAC, which corresponds to a weak security assumption on the FIL primitive in this context. Let us begin by recalling some background. We then describe more precisely the problem we consider, its motivation and history, and our results.

1.1 Background

MACs. Recall that a message authentication code (MAC) is the most common mechanism for assuring integrity of data communicated between parties who share a secret key k. A MAC is specified by a function q that takes the key kand data x to produce a tag $\tau = g(k, x)$. The sender transmits (x, τ) and the receiver verifies that $g(k,x) = \tau$. The required security property is unforgeability, namely that even under a chosen-message attack, it be computationally infeasible for an adversary (not having the secret key k) be able to create a valid pair (x,τ) which is "new" (meaning x has not already been authenticated by the legitimate parties). As the main tool for ensuring data integrity and access control, much effort goes into the design of (secure and efficient) MACs, and many constructions are known. These include block cipher based MACs like the CBC MAC [2] or XOR MACs [9]; hash function based MACs like HMAC [3] or MDx-MAC [20]; and universal hash function based MACs [10,23]. Many of the existing constructions of MACs fall into the category of FIL to VIL transforms. For example the CBC MAC iterates applications of a block cipher (the underlying FIL primitive), while hash function based MACs iterate (implicitly or explicitly) applications of the underlying compression function.

ASSUMPTIONS UNDERLYING THE TRANSFORMS. Analyses of existing block cipher based MACs make stronger assumptions on the underlying FIL primitive than the unforgeability required of the final VIL-MAC. For example, security analyses of the CBC or XOR MACs provided in [6,9] model the underlying block cipher as a pseudorandom function, assumed to be "unpredictable" in the sense of [12], a requirement more stringent than unforgeability.

The security analysis of HMAC¹ provided in [3] makes two assumptions: that the (appropriately keyed) compression function is a MAC and also that the iterated compression function is "weakly collision resistant". Thus, the security of HMAC is not shown to follow from an assumption only about the underlying FIL primitive.

Universal hash function based MACs don't usually fall in the FIL to VIL paradigm, but on the subject of assumptions one should note that they require the use of block ciphers modeled as pseudorandom functions to mask the output of the (unconditionally secure) universal hash function, and thereby use assumptions stronger than unforgeability on the underlying primitives.

1.2 From FIL-MACs to VIL-MACs

THE PROBLEM. We are interested in obtaining VIL-MACs whose security can be shown to follow from (only) the assumption that the underlying FIL primitive is itself a MAC. In other words, we wish to stay within the standard paradigm of transforming a FIL primitive to a VIL-MAC, but we wish the analysis to make a minimal requirement on the security of the given FIL primitive: it need not be unpredictable, but need only be a MAC itself, namely unforgeable. This

To be precise, the security analysis we refer to is that of NMAC, of which HMAC is a variant.

is, we feel, a natural and basic question, yet one that surprisingly has not been systematically addressed.

Benefits of reduced assumptions. It is possible that an attack against the pseudorandomness of a block cipher may be found, yet not one against its unforgeability. A proof of security for a block cipher based MAC that relied on the pseudorandomness assumption is then rendered void. (This does not mean there is an attack on the MAC, but it means the MAC is not backed by a security guarantee in terms of the cipher.) If, however, the proof of security had only made an unforgeability assumption, it would still stand and lend a security guarantee to the MAC construction. Similarly, collision-resistance of a compression function might be found to fail, but the unforgeability of some keyed version of this function may still be intact. (This is true for example for the compression function of MD5.) Thus, if the security analysis of a (keyed) compression-function based MAC relied only on an unforgeability assumption, the security guarantee on the MAC would remain.

Another possibility enabled by this approach would be to design FIL-MACs from scratch. Since the security requirement is weaker than for block ciphers, we might be able to get FIL-MACs that are faster than block ciphers, and thereby speed up message authentication.

1.3 Our Results

The benefit (of a VIL-MAC with a security analysis relying only on the assumption that the FIL primitive is a MAC) would be greatest if the construction were an existing, in use one, whose security could now be justified under a weaker assumption. In that case, existing MAC implementations could be left unmodified, but benefit from an improved security guarantee arising from relying only on a weaker assumption. Accordingly, we focus on existing transforms (or slight variants) and ask whether they preserve unforgeability.

CBC MAC. The first and most natural candidate is the CBC MAC. Recall that given a FIL primitive $f: \{0,1\}^{\kappa} \times \{0,1\}^{l} \to \{0,1\}^{l}$ its CBC MAC is the transform CBC[f], taking key $k \in \{0,1\}^{\kappa}$ and input $x = x_1 \dots x_n \in \{0,1\}^{ln}$ to return y_n , where $y_i = f(k, y_{i-1} \oplus x_i)$ for $1 \le i \le n$, and $y_0 = 0^l$. We already know that if f is a pseudorandom function then CBC[f] is a secure MAC [f], and the question is whether the assumption that f itself is only a MAC is enough to prove that CBC[f] is a secure MAC. We show that it is not. We do this by exhibiting a f that is a secure MAC, but for which there is an attack showing that CBC[f] is not a secure MAC. (This relies of course on the assumption that some secure FIL-MAC exists, since otherwise the question is void.)

MD METHOD. Next we look at Damgård's method [11] for transforming a keyed compression function $f: \{0,1\}^{\kappa} \times \{0,1\}^{\ell+b} \to \{0,1\}^{\ell}$ into a full-fledged hash function.² Actually our method differs slightly in the way it handles input-length

² The construction of Damgård is essentially the same as that of Merkle, except that in the latter, the given compression function is keyless, while in the former, it is keyed. Since MACs are keyed, we must use Damgård's setting here.

variability, which it does by using another key. Our nested, iterated construction, NI[f], takes keys k_1, k_2 and input $x = x_1 \dots x_n \in \{0, 1\}^{nb}$ to return $f_{k_2}(y_n ||\langle |x| \rangle)$, where $y_i = f_{k_1}(y_{i-1}||x_i)$ for $1 \le i \le n$ and $y_0 = 0^{\ell}$ and $\langle |x| \rangle$ is the length of x written as a binary string of length exactly b bits.

Although the construction is (essentially) the one used in the collision-resistant hash setting, the analysis needs to be different. This is because of two central differences between MACs and hash functions: MACs rely for their security on a secret key, while hash functions (which, in the Damgård setting, do use a key) make this key public; and the security properties in question are different (unforgeability for MACs, and collision-resistance for hash functions).

We show that if f is a secure MAC then so is NI[f]. The analysis has several steps. As an intermediate step in the analysis we use the notion of weak-collision resistance of [3], and one of our lemmas provides a connection between this and unforgeability.

An appropriately keyed version of the compression function of any existing cryptographic hash function can play the role of f above, as illustrated in Section 4.1. This provides another solution to the problem of using keyed compression functions to design MACs. In comparison with HMAC, the nested, iterated construction has lower throughput because each iteration of the compression function must use a key. Implementation also requires direct access to the compression function, as opposed to being implementable only by calls to the hash function itself. On the other hand, the loss in performance is low, it is still easy to implement, and the supporting security analysis makes weaker assumption than that of HMAC.

FEISTEL. The Feistel transform is another commonly used method of increasing the amount of data one can process with a given FIL primitive. The basic transform doubles the input length of a given function f. The security of this transform as a function of the number of rounds r has been extensively analyzed for the problem of transforming a pseudorandom function into a pseudorandom permutation: Luby and Rackoff [15] showed that two rounds do not suffice for this purpose, but three do. We ask whether r rounds of Feistel on a MAC f result in a MAC. The answer is easily seen to be no for r=2. But we also show that it remains no for r=3, meaning that the 3-round Feistel transform that turns pseudorandom functions into pseudorandom permutations does not preserve unforgeability. Furthermore, even more rounds do not appear to help in this regard.

1.4 Related Work

The FIL to VIL question that we address for MACs is an instance of a classic one, which has been addressed before for many other primitives and has played an important role in the development of the primitives in question. The attraction of the paradigm is clear: It is easier to design and do security analyses for the "smaller", FIL primitives, and then build the VIL primitive on top of them.

The modes of operation of block ciphers were probably the earliest constructions in this area, but an analysis in the light of this paradigm is relatively recent [5]. Perhaps the best known example is the Merkle-Damgård [16,11] iteration method used in the case of collision-resistant functions. Another early example is (probabilistic) public-key encryption, where Goldwasser and Micali showed that bit-by-bit encryption of a message preserves semantic security [13]. (The FIL primitive here is encryption of a single bit.) Extensive effort has been put into this problem for the case of pseudorandom functions (the problem is to turn a FIL pseudorandom function into a VIL one) with variants of the CBC (MAC) construction [6,19] and the cascade construction [4] being solutions. Bellare and Rogaway considered the problem and provided solutions for TCR (target-collision-resistant) hashing [7], a notion of hashing due to Naor and Yung [18] which the latter had called universal one-way hashing.

Curiously, the problem of transforming FIL-MACs to VIL-MACs has not been systematically addressed prior to our work. However, some constructions are implicit. Specifically, Merkle's hash tree construction [17] can be analyzed in the case of MACs. Bellare, Goldreich and Goldwasser use such a design to build incremental MACs [8], and thus a result saying that the tree design transforms FIL-MACs to VIL-MACs seems implicit here.

2 Definitions

FAMILIES OF FUNCTIONS. A family of functions is a map $F: Keys(F) \times Dom(F) \to Rng(F)$, where Keys(F) is the set of keys of F; Dom(F) is some set of input messages associated to F; and Rng(F) is the set of output strings associated to F. For each key $k \in Keys(F)$ we let $F_k(\cdot) = F(k, \cdot)$. This is a map from Dom(F) to Rng(F). If $Keys(F) = \{0, 1\}^{\kappa}$ for some κ then the latter is the key-length. If $Dom(F) = \{0, 1\}^{\delta}$ for some δ then δ is called the input length.

MACs. A MAC is a family of functions F. It is a FIL-MAC (fixed-input-length MAC) if Dom(F) is $\{0,1\}^b$ for some small constant b, and it is a VIL-MAC (variable input length MAC) if Dom(F) contains strings of many different lengths. The security of a MAC is measured via its resistance to existential forgery under chosen-message attack, following [6], which in turn is a concrete security adaptation to the MAC case of the notion of security for digital signatures of [14]. We consider the following experiment Forge(A, F) where A is an adversary (forger) who has access to an oracle for $F_k(\cdot)$:

```
Experiment Forge(A, F)

k \leftarrow Keys(F); (m, \tau) \leftarrow A^{F_k(\cdot)}

If F_k(m) = \tau and m was not an oracle query of A

then return 1 else return 0
```

We denote by $\mathbf{Succ}_F^{\mathrm{mac}}(A)$ the probability that the outcome of the experiment $\mathrm{Forge}(A,F)$ is 1. We associate to F its insecurity function, defined for any integers t,q,μ by

$$\mathbf{InSec}_F^{\mathrm{mac}}(t,q,\mu) \, \stackrel{\mathrm{def}}{=} \, \max_{A} \, \{ \, \mathbf{Succ}_F^{\mathrm{mac}}(A) \, \} \; .$$

Here the maximum is taken over all adversaries A with "running time" t, "number of queries" q, and "total message length" μ . We put the resource names in quotes because they need to be properly defined, and in doing so we adopt some important conventions. Specifically, resources pertain to the experiment Forge(A, F) rather than the adversary itself. The "running time" of A is defined as the time taken by the experiment $\operatorname{Forge}(A, F)$ (we call this the "actual running time") plus the size of the code implementing algorithm A, all this measured in some fixed RAM model of computation. We stress that the actual running time includes the time of all operations in the experiment Forge(A, F); specifically it includes the time for key generation, computation of answers to oracle queries, and even the time for the final verification. To measure the cost of oracle queries we let Q_A be the set of all oracle queries made by A, and let $Q = Q_A \cup \{m\}$ be union of this with the message in the forgery. Then the number of queries q is defined as |Q|, meaning m is counted (because of the verification query involved). Note also that consideration of these sets means a repeated query is not double-counted. Similarly the total message length is the sum of the lengths of all messages in Q. These conventions will simplify the treatment of concrete security.

The insecurity function is the maximum likelihood of the security of the message authentication scheme F being compromised by an adversary using the indicated resources. We will speak informally of a "secure MAC"; this means a MAC for which the value of the insecurity function is "low" even for "reasonably high" parameter values. When exactly to call a MAC secure is not something we can pin down ubiquitously, because it is so context dependent. So the term secure will be used only in discussion, and results will be stated in terms of the concrete insecurity functions.

3 The CBC MAC Does not Preserve Unforgeability

Let $f: \{0,1\}^{\kappa} \times \{0,1\}^{l} \to \{0,1\}^{l}$ be a family of functions. For any fixed integer n > 0 we define the associated CBC MAC. It is the family of functions $CBC[f]: \{0,1\}^{\kappa} \times \{0,1\}^{ln} \to \{0,1\}^{l}$ defined as follows:

```
Algorithm CBC[f](k, x_1 ... x_n)

y_0 \leftarrow 0^l

For i = 1, ..., n do y_i \leftarrow f(k, y_{i-1} \oplus x_i)

Return y_n
```

Here $k \in \{0,1\}^{\kappa}$ is the key, and x_i is the *i*-th *l*-bit block of the input message. We know that if f is a pseudorandom function then CBC[f] is a secure MAC [6]. Here we show that the weaker requirement that f itself is only a secure MAC does not suffice to guarantee that CBC[f] is a secure MAC. Thus, the security of the CBC MAC needs relatively strong assumptions on the underlying primitive.

We stress that the number of message blocks n is fixed. If not, splicing attacks are well-known to break the CBC MAC. But length-variability can be dealt with

in a variety of ways (cf. [6,19]), and since the results we show here are negative, they are only strengthened by the restriction to a fixed n.

We prove our claim by presenting an example of a MAC f which is secure, but for which we can present an attack against CBC[f]. We construct f under the assumption that some secure MAC exists, since otherwise there is no issue here at all.

Assume we have a secure MAC $g: \{0,1\}^{\kappa} \times \{0,1\}^{2m} \to \{0,1\}^{m}$ whose input length is twice its output length. We set l=2m and transform g into another MAC $f: \{0,1\}^{\kappa} \times \{0,1\}^{l} \to \{0,1\}^{l}$. We show that f is a secure MAC but CBC[f] is not. Below we present f as taking a κ -bit key k and an l-bit input $a=a_1||a_2|$ which we view as divided into two m-bit halves.

```
Algorithm f(k, a_1 a_2)

\sigma \leftarrow g(k, a_1 a_2)

Return \sigma a_1
```

That is, f_k on any input simply returns g_k on the same input, concatenated with the first half of f_k 's input. It should be clear intuitively that f is a secure MAC given that g is a secure MAC, because the output of f contains a secure MAC on the input, and the adversary already knows the data a_1 anyway. The following claim relates the securities more precisely. Its proof can be found in [1].

```
Claim. Let g, f be as above. Then \mathbf{InSec}_f^{\mathrm{mac}}(t, q, \mu) \leq \mathbf{InSec}_q^{\mathrm{mac}}(t, q, \mu).
```

We now show that the CBC MAC method is not secure if we use the function f as the underlying base function. The following claim says that there is an attack on $\mathrm{CBC}[f]$, which after obtaining the correct tag of only one chosen message, succeeds in forging the tag of a new message. The attack is for the case n=2 of two block messages, so that both the chosen message and the one whose tag is forged have length 2l.

Claim. There is a forger F making a single 2l-bit query to $CBC[f](k,\cdot)$ and achieving $Succ^{mac}_{CBC[f]}(F) = 1$.

Proof. The attacker F is given an oracle for $\mathrm{CBC}[f](k,\cdot)$ and works as follows:

```
Forger F^{\text{CBC}[f](k,\cdot)}

Let a_1, a_2 be distinct m-bit strings and let x \leftarrow a_1 a_2 0^m 0^m

\sigma_2 \sigma_1 \leftarrow \text{CBC}[f](k, x)

x_1' \leftarrow a_1 a_2 \; ; \; x_2' \leftarrow a_1 a_2 \oplus \sigma_1 a_1 \; ; \; x' \leftarrow x_1' x_2'

Return (x', \sigma_1 a_1)
```

Here F first defined the 2l bit message x. It then obtained its l-bit tag from the oracle, and split it into two halves. It then constructed the l-bit blocks x'_1, x'_2 as shown, and concatenated them to get x', which it output along with the claimed tag $\sigma_1 a_1$.

To show that this is a successful attack, we need to check two things. First that the forgery is valid, meaning $CBC[f](k, x') = \sigma_1 a_1$, and second that the message x' is new, meaning $x' \neq x$.

Let's begin with the second. We note that the last m bits of x' are $a_2 \oplus a_1$. But F chose a_1, a_2 so that $a_2 \neq a_1$ so $a_2 \oplus a_1 \neq 0^m$. But the last m bits of x are zero. So $x' \neq x$.

Now let us verify that $CBC[f](k, x') = \sigma_1 a_1$. By the definition of f in terms of g, and by the definition of CBC[f], we have

$$CBC[f](k, x) = f(k, f(k, a_1 a_2) \oplus 0^m 0^m)$$

= $f(k, g(k, a_1 a_2) a_1)$
= $g(k, g(k, a_1 a_2) a_1) || g(k, a_1 a_2) .$

This implies that $\sigma_1 = g(k, a_1 a_2)$ and $\sigma_2 = g(k, \sigma_1 a_1)$ in the above code. Using this we see that

$$CBC[f](k, x') = f(k, f(k, a_1 a_2) \oplus (a_1 a_2 \oplus \sigma_1 a_1))$$

$$= f(k, \sigma_1 a_1 \oplus (a_1 a_2 \oplus \sigma_1 a_1))$$

$$= f(k, a_1 a_2)$$

$$= \sigma_1 a_1$$

as desired. \Box

The construct f above that makes the CBC MAC fail is certainly somewhat contrived; indeed it is set up to make the CBC MAC fail. Accordingly, one reaction to the above is that it does not tell us anything about the security of, say, DES-CBC, because DES does not behave like the function f above. This reaction is not entirely accurate. The question here is whether the assumption that the underlying cipher is a MAC is sufficient to be able to prove that its CBC is also a MAC. The above says that no such proof can exist. So with regard to DES-CBC, we are saying that its security relies on stronger properties of DES than merely being a MAC, for example pseudorandomness.

4 The NI Construction Preserves Unforgeability

Here we define the nested, iterated transform of a FIL-MAC and show that the result is a VIL-MAC.

4.1 The Construction

We are given a family of functions $f: \{0,1\}^{\kappa} \times \{0,1\}^{\ell+b} \to \{0,1\}^{\ell}$ which takes the form of a (keyed) compression function, and we will associate to this the nested iterated (NI) function NI[f]. The construction is specified in two steps; we first define the iteration of f and then show how to get NI[f] from that. See Figure 1 for the pictorial description.

CONSTRUCTION. As the notation indicates, the input to any instance function $f(k,\cdot)$ of the given family has length $\ell+b$ bits. We view such an input as divided into two parts: a *chaining variable* of length ℓ bits and a data block of length ℓ bits. We associate to f its *iteration*, a family $\mathrm{IT}[f] \colon \{0,1\}^{\kappa} \times \{0,1\}^{\leq L} \to \{0,1\}^{\kappa}$

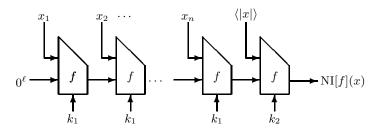


Fig. 1. The nested, iterated construction of a VIL-MAC given a FIL-MAC f.

 $\{0,1\}^{\ell+b}$, where L is to be defined, and for any key k and string x of length at most L we define:

```
Algorithm IT[f](k, x)

y_0 \leftarrow 0^{\ell}

Break x into b-bit blocks, x = x_1 \dots x_n

For i = 1, \dots, n do y_i \leftarrow f(k, y_{i-1} || x_i)

a \leftarrow y_n ||\langle |x| \rangle

Return a
```

Above if |x| is not a multiple of b, some appropriate padding mechanism is used to extend it. By $\langle |x| \rangle$ we denote a binary representation of |x| as a string of exactly b bits. This representation is possible as long as $|x| < 2^b$, and so we set the maximum message length to $L = 2^b - 1$. This is hardly a restriction in practice given that typical values of b are large.

Now we define the family NI[f]: $\{0,1\}^{2\kappa} \times \{0,1\}^{\leq L} \to \{0,1\}^{\ell}$. A key for this family is a pair k_1k_2 of κ -bit keys, and for a string x of length at most L we set:

```
Algorithm NI[f](k_1k_2, x)

a \leftarrow \text{IT}[f](k_1, x)

Return f(k_2, a)
```

RELATION TO OTHER CONSTRUCTS. Our f has the syntactic form of a (keyed) compression function. The quantity y_n computed in the code of $\mathrm{IT}[f]$ is obtained via the iteration method of Damgård [11]; our iterated function is different only in that it appends to this the length of the input x. The main difference is in the security properties. Whereas Damgård assumes f is collision-resistant and wants to show that $\mathrm{IT}[f]$ is too, we assume f is a FIL-MAC and want to show $\mathrm{NI}[f]$ is a VIL-MAC. The difference is that for MACs the key is secret while for hash functions it is public, and the notions of security are not the same.

Preneel and Van Oorschot [20] suggest that in designing MACs from iterated hash functions, one might use a keyed compression function with a secret key and keyed output transformation. Modulo the handling of length-variability, this is exactly our construction. Preneel et. al. however did not analyze this construction under the assumption that the compression function is a MAC.

Comparing our construction to HMAC/NMAC, the difference, roughly speaking, is that HMAC is based on a hash function (like MD5 or SHA-1) that uses a compression function that is keyless, and iterated in the Merkle style [16]. Had we instead started with a hash function that iterated a keyed compression function in the Damgård style, and applied the HMAC transform to it, we would end up with essentially our construction. This tells us that the Damgård's setting and construction have a nice extra feature not highlighted before: they adapt to the MAC setting in a direct way.

Another difference between our construction and NMAC lies in how the output of the internal functions of the nested functions are formed. Our internal function IT[f] appends the length of the message and the appended length is a part of the function's output whereas F (in NMAC) applies the base function once more on the length of the message.

Instantiation. Appropriately keying the compression function of some existing cryptographic hash function will yield a candidate for f above. For example, let sha-1: $\{0,1\}^{160+512} \rightarrow \{0,1\}^{160}$ be the compression function of SHA-1. We can key it via its 160-bit chaining variable. We would then use the 512 bit regular input as the input of the keyed function. This means we must further subdivide it into two parts, one to play the role of a new chaining variable and another to be the actual data input. This means we set $\kappa = \ell = 160$ and b = 352, and define the keyed sha-1 compression function ksha-1: $\{0,1\}^{160} \times \{0,1\}^{160+352} \rightarrow \{0,1\}^{160}$ by

$$ksha-1(k, a||b) = sha-1(k||a||b),$$

for any key $k \in \{0,1\}^{160}$, any $a \in \{0,1\}^{160}$ and any $b \in \{0,1\}^{352}$. Now, we can implement NI[ksha-1] and this will be a secure MAC under the assumption that ksha-1 was a secure MAC on 352 bit messages.

Note that under this instantiation, each application of sha-1 will process 352 bits of the input, as opposed to 512 in a regular application of sha-1 as used in SHA-1 or HMAC-SHA-1. So the throughput of NI[ksha-1] is a factor of $352/512 \approx 0.69$ times that of HMAC-SHA-1. Also, implementation of NI[ksha-1] calls for access to sha-1; unlike HMAC-SHA-1, it cannot be implemented by calls only to SHA-1. On the other hand, the security of NI[ksha-1] relies on weaker assumptions than that of HMAC-SHA-1. The analysis of the latter assumes that ksha-1 is a secure MAC and that the iteration of sha-1 is weakly collision-resistant; the analysis of NI[ksha-1] makes only the former assumption.

4.2 Security Analysis

Our assumption is that f above is a secure FIL-MAC. The following theorem says that under this condition (alone) the nested iterated construction based on f is a secure VIL-MAC. The theorem also indicates the concrete security of the transform.

Theorem 1. Let $f: \{0,1\}^{\kappa} \times \{0,1\}^{\ell+b} \to \{0,1\}^{\ell}$ be a fixed input-length MAC. Then the nested, iterated function family $NI[f]: \{0,1\}^{2\kappa} \times \{0,1\}^{\leq L} \to \{0,1\}^{\ell}$ is

a variable input-length MAC with

$$\mathbf{InSec}^{\mathrm{mac}}_{\mathrm{NI}[f]}(t,q,\mu) \leq \left(1 + \frac{\mu}{b}\right)^2 \cdot \mathbf{InSec}^{\mathrm{mac}}_f(t',q',\mu')$$
where $t' = t + O(\mu')$, $q' = \mu/b$, and $\mu' = (b + \ell) \cdot \mu/b$.

TIGHTNESS OF THE BOUND. There is an appreciable loss in security above, with the insecurity of the nested iterated construct being greater than that of the original f by a factor of (roughly) the square of the number μ/b of messages in a chosen-message attack on f. This loss in security is however unavoidable. Iterated constructs of this nature continue to be subject to the birthday attacks illustrated by Preneel and Van Oorschott [20], and these attacks can be used to show that the above bound is essentially tight.

PROOF APPROACH. A seemingly natural approach to proving Theorem 1 would be to try to imitate the analyses of Merkle and Damgård [16,11] which showed that transforms very similar to ours preserve collision-resistance. This approach however turned out to be less straightforward to implement here than one might imagine, due to our having to deal with forgeries rather than collisions. Accordingly we take a different approach. We first reduce the question of forgeries to one about a certain kind of collision-resistance, namely "weak-collision resistance", showing that the insecurity of our construct as a MAC can be bounded in terms of its weak collision-resistance and the insecurity of the original f as a MAC. We can bound the weak collision-resistance of the iterated construct in terms of the weak collision-resistance of the original f using the approach of [16,11], and finally bound the weak-collision resistance of f in terms of its insecurity as a MAC. Putting the three together yields the theorem.

Underlying many of these steps are general lemmas, and we state them in their generality since they might be of independent interest. In particular, we highlight the connections between weak collision-resistance and MACs. We need to begin, however, by saying what is weak collision-resistance.

Weak Collision-resistance. In the usual attack model for finding collisions, the adversary is able to compute the hash function for which it seeks collisions; either it is a single, public function, or, if a family F, the key k (defining the map F_k for which the adversary seeks collisions) is given to the adversary. In the weak collision-resistance setting as defined in [3], the adversary seeking to find collisions for F_k is not given k, but rather has oracle access to F_k . Weak collision-resistance is thus a less stringent requirement than standard collision-resistance.

Let $F: \{0,1\}^{\kappa} \times Dom(F) \to Rng(F)$ be a family of functions. To formally define its weak collision-resistance we consider the following experiment. Here A is an adversary that gets an oracle for F_k and returns a pair of points m, m' in Dom(F). It wins if these points are a (non-trivial) collision for F_k .

```
Experiment FindWeakCol(A, F)

k \leftarrow Keys(F); (m, m') \leftarrow A^{F_k(\cdot)}

If m \neq m' and F_k(m) = F_k(m') then return 1 else return 0
```

We denote by $\mathbf{Succ}_F^{\mathrm{wcr}}(A)$ the probability that the above experiment returns 1. We then define

$$\mathbf{InSec}_F^{\mathrm{wcr}}(t,q,\mu) \, \stackrel{\mathrm{def}}{=} \, \max_{A} \, \{ \, \mathbf{Succ}_F^{\mathrm{wcr}}(A) \, \} \; .$$

As before the maximum is taken over all adversaries A with "running time" t, "number of queries" q, and "total message length" μ , the quantities in quotes being measured with respect to the experiment FindWeakCol(A, F), analogously to the way they were measured in the definition of $\mathbf{InSec}^{\mathrm{mac}}$ described in Section 2. Specifically the running time is the actual execution time of FindWeakCol(A, F) plus the size of the code of A. We let $Q = Q_A \cup \{m, m'\}$ where Q_A is the set of all queries made by A. Then q = |Q| and μ is the sum of the lengths of all messages in Q.

REDUCTION TO WCR. We bound the insecurity of the nested construct as a MAC in terms of its weak-collision resistance and MAC insecurity of the original function. The following generalizes and restates a theorem on NMAC from [3]. In our setting h will be IT[f], and then N becomes NI[f].

Lemma 1. Let $f: \{0,1\}^{\kappa} \times \{0,1\}^{\ell+b} \to \{0,1\}^{\ell}$ be a fixed input-length MAC, and let $h: \{0,1\}^{\kappa} \times D \to \{0,1\}^{\ell+b}$ be a weak collision-resistant function family on some domain D. Define $N: \{0,1\}^{2\kappa} \times D \to \{0,1\}^{\ell}$ via

$$N(k_1k_2, x) = f(k_2, h(k_1, x))$$

for any keys $k_1, k_2 \in \{0, 1\}^{\kappa}$ and any $x \in D$. Then N is a MAC with

$$\mathbf{InSec}_N^{\mathrm{mac}}(t,q,\mu) \leq \mathbf{InSec}_f^{\mathrm{mac}}(t,q,q(b+\ell)) + \mathbf{InSec}_h^{\mathrm{wcr}}(t,q,\mu)$$

for all t, q, μ .

The proof of the above is an adaptation of the proof in [3] which is omitted here, but for completeness is provided in [1].

To prove Theorem 1 we will apply the above lemma with h = IT[f]. Accordingly our task now reduces to bounding the weak collision-resistance insecurity of IT[f]. But remember that we want this bound to be in terms of the insecurity of f as a MAC. We thus obtain the bound in two steps. We first bound the weak collision-resistance of IT[f] in terms of the weak collision-resistance of f, and then bound the latter via its insecurity as a MAC.

WEAK COLLISION-RESISTANCE OF IT[f]. We now show that if f is a weak collision-resistant function family, then the iterated construction IT[f] is also a weak collision-resistant function family.

Lemma 2. Let $f: \{0,1\}^{\kappa} \times \{0,1\}^{\ell+b} \to \{0,1\}^{\ell}$ be a weak collision-resistant function family. Then,

$$\mathbf{InSec}^{\mathrm{wcr}}_{\mathrm{IT}[f]}(t,q,\mu) \leq \mathbf{InSec}^{\mathrm{wcr}}_f(t,\frac{\mu}{b},(b+\ell)\frac{\mu}{b})$$

The proof is analogous to those in [16,11] which analyze similar constructs with regard to (standard, not weak) collision-resistance. To extend them one must first observe that their reductions make only black-box use of the underlying

function instances and can thus be implemented in the weak collision-resistance setting via the oracle for the function instance. Second, our way of handling the length variability, although different, can be shown to work. The proof is omitted here, but for completeness is provided in [1].

Given the above two lemmas our task has reduced to bounding the weak collision-resistance insecurity of f in terms of its MAC insecurity. The connection is actually much more general.

Weak Collision-Resistance of any MAC. We show that any secure MAC is weakly collision-resistant, although there is a loss in security in relating the two properties. This is actually the main lemma in our proof, and may be of general interest.

Lemma 3. Let $g: \{0,1\}^{\kappa} \times Dom(g) \to \{0,1\}^{\ell}$ be a family of functions. Then, $\mathbf{InSec}_{g}^{\mathrm{wcr}}(t,q,\mu) \leq q^{2} \cdot \mathbf{InSec}_{g}^{\mathrm{mac}}(t+O(\mu),q,\mu)$

The proof of the above is given in Appendix A.

PROOF OF THEOREM 1. We now use the three lemmas above to complete the proof of Theorem 1. Letting $\epsilon = \mathbf{InSec}_f^{\mathrm{mac}}(t, q, q(b+\ell))$ for conciseness, we have:

$$\mathbf{InSec}^{\mathrm{mac}}_{\mathrm{NI}[f]}(t,q,\mu)$$

$$\leq \epsilon + \mathbf{InSec}_{\mathrm{IT}[f]}^{\mathrm{wcr}}(t, q, \mu)$$
 (1)

$$\leq \epsilon + \mathbf{InSec}_f^{\mathrm{wcr}}(t, \frac{\mu}{b}, (b+\ell)\frac{\mu}{b})$$
 (2)

$$\leq \epsilon + \left(\frac{\mu}{b}\right)^2 \mathbf{InSec}_f^{\mathrm{mac}}(t + O((b+\ell)\frac{\mu}{b}), \frac{\mu}{b}, (b+\ell)\frac{\mu}{b}) \tag{3}$$

$$= \left(1 + \frac{\mu}{b}\right)^2 \cdot \mathbf{InSec}_f^{\mathrm{mac}}(t', q', \mu') , \qquad (4)$$

where $t' = t + O(\mu')$, $q' = \mu/b$ and $\mu' = (b + \ell) \cdot \mu/b$. In Equation 1 we used Lemma 1. In Equation 2 we used Lemma 2. In Equation 3, Lemma 3 is used, and the two terms of $\mathbf{InSec}_f^{\mathrm{mac}}$ are added in Equation 4 with the larger of the two resource parameters taken as the final resource parameters to obtain the conclusion of the theorem.

5 Feistel Does not Preserve Unforgeability

Let $f: \{0,1\}^{\kappa} \times \{0,1\}^l \to \{0,1\}^l$ be a family of functions. For any fixed integer r > 0, we define the r-round Feistel transform. It is a family of functions $\text{FST}^r[f]: \{0,1\}^{r\kappa} \times \{0,1\}^{2l} \to \{0,1\}^{2l}$. Given keys k_1,\ldots,k_r and input LR where |L| = |R| = l, we define

Algorithm $\operatorname{FST}^r[f](k_1 \dots k_r, LR)$ $L_0 \leftarrow L \; ; \; R_0 \leftarrow R$ For $i = 1, \dots, r \text{ do } Z_{i-1} \leftarrow f_{k_i}(R_{i-1}) \; ; \; R_i \leftarrow L_{i-1} \oplus Z_{i-1} \; ; \; L_i \leftarrow R_{i-1}$ Return $L_r R_r$ Here $L_i R_i$ is the 2*l*-bit block at the end of the *i*-th round. The Feistel transform has been used extensively to extend (double) the input size of a given pseudorandom function. Luby and Rackoff have shown that $FST^3[f]$ is a pseudorandom permutation if f is a pseudorandom function [15]. Here we examine the possibility of $FST^r[f]$ being a secure MAC under the assumption that f is only a secure MAC.

Luby and Rackoff showed that $FST^2[f]$ is not pseudorandom even if f is pseudorandom [15]. It is easy to design an attack showing that $FST^2[f]$ is not a secure MAC even if f is a secure MAC. (This is provided in [1] for completeness.) Here we go on to the more interesting case of three rounds, where the transform is known to be pseudorandomness preserving.

We show however that the three round Feistel transform does not preserve unforgeability. Namely the assumption that f is a secure MAC does not suffice to guarantee that $\text{FST}^3[f]$ is a secure MAC. We prove our claim by presenting an attack against $\text{FST}^3[f]$ when f is the MAC of Section 3 for which we had presented an attack against CBC[f]. Recall that $f \colon \{0,1\}^{\kappa} \times \{0,1\}^{l} \to \{0,1\}^{l}$ was designed in terms of an underlying secure but arbitrary MAC $g \colon \{0,1\}^{2m} \to \{0,1\}^{m}$, and we set l=2m. Let us now see what happens when we evaluate $\text{FST}^3[f](k_1k_2k_3, L_0R_0)$. We write $L_0=a_0\|a_1$ and $R_0=b_0\|b_1$ where $|a_0|=|a_1|=|b_0|=|b_1|=m=l/2$ bits, and work through the three Feistel rounds, writing the intermediate results in terms of the notation used in describing the Feistel algorithm above:

Here we have set

$$\mu = g(k_1, b_0 b_1)$$

$$\mu' = g(k_2, a_0 \oplus \mu || a_1 \oplus b_0)$$

$$\mu'' = g(k_3, b_0 \oplus \mu' || b_1 \oplus a_0 \oplus \mu).$$

Write $L_3 = a_3 \| a_3'$ and $R_3 = b_3 \| b_3'$. We notice that given the output $L_3 R_3$ and the input $L_0 R_0$, it is possible to extract the values μ, μ', μ'' , even without knowledge of any of the keys. Namely $\mu = b_1 \oplus a_0 \oplus a_3'$ and $\mu' = a_3 \oplus b_0$ and $\mu'' = a_0 \oplus \mu \oplus b_3$. Furthermore notice that once an attacker has these values, it can also compute Z_0, Z_1, Z_2 , the internal Feistel values. Based on this we will present an attack against against FST³[f].

Claim. There is a forger A making four 2l-bit queries to $\text{FST}^3[f](k,\cdot)$ and achieving $\mathbf{Succ}^{\text{mac}}_{\text{FST}^3[f]}(A) = 1 - O(2^{-l})$.

Proof. The attacker A is given an oracle for $\mathrm{FST}^3[f](k,\cdot)$, where $k=k_1k_2k_3\in\{0,1\}^{3\kappa}$ is the key. It makes the four queries displayed, respectively, as the first rows of the first four columns in Figure 2. The first two queries generated by A are random. A then generates the next two queries adaptively, using the results of the previous queries. Notice that the third and fourth queries are functions of Z-values occurring in the 3-round Feistel computation on the first two queries. The attacker A can obtain these values using the observation above. Finally, A comes up with the forgery (x,τ) , where x and τ are displayed, respectively, as the first and last rows in the fifth column of the same Figure.

query	1 query 2	query 3	query 4	forgery
$L_0^1 \mid F$	$L_0^1 L_0^2 \mid R_0^2$	$L_0^3 \mid R_0^2 \oplus Z_1^1 \oplus Z_1^2$	$R_1^2 \oplus Z_0^3 \mid R_0^2 \oplus Z_1^1 \oplus Z_1^2$	$R_1^1 \oplus Z_0^2 \mid R_0^2$
2	$ Z_0^1 Z_0^2 $	$ Z_0^3 $	$\mid Z_0^3$	$ Z_0^2 $
$R_0^1 \mid F$	$R_1^1 R_0^2 \mid R_1^2$		$R_0^2 \oplus Z_1^1 \oplus Z_1^2 \mid R_1^2$	$R_0^2 \mid R_1^1$
2	$ Z_1^1 Z_1^2 $		$\mid Z_1^2$	$\mid Z_1^1$
$R_1^1 \mid F$	$R_2^1 \mid R_1^2 \mid R_2^2$		$R_1^2 \mid R_0^2 \oplus Z_1^1$	$R_1^1 \mid R_0^2 \oplus Z_1^1$
2	$ Z_2^1 Z_2^2 $		$\mid Z_2^3$	$\mid Z_2^3$
$R_2^1 \mid F$	$R_3^1 R_2^2 \mid R_3^2$			$R_0^2 \oplus Z_1^1 \mid R_1^1 \oplus Z_2^3$

Fig. 2. Contents of the queries and the intermediate/final results and the forgery.

Notice that in queries 3 and 4 in Figure 2, the rows after certain values (Z_0^3, Z_2^3) are empty. They are omitted for simplicity because only those two values (Z_0^3, Z_2^3) are needed to form the next queries or the forgery, and the rest of the values are not needed. In the actual attack, those values are computed from the outputs of the oracle.

To show that this is a successful attack, we need to check two things. First that the forgery is valid, meaning $FST^3[f](k,x) = \tau$, and second that the message x is new, meaning $x \notin \{x_1 \dots x_4\}$.

We can easily see that the forgery is valid by examining the values in the table. The second requirement that x is new can be achieved with high probability if the adversary chooses the strings L_0^1 , L_0^2 , and L_0^3 randomly. If the said strings are chosen randomly, then the l-bit left-half of each queried string becomes random and the probability of the forgery string x matching any one of the four queried strings is very small $(O(2^{-l}))$. This means that the probability of the forgery being new is $1 - O(2^{-l})$ as shown in the claim. Hence, the above attack succeeds with high probability $(1 - O(2^{-l}))$ as desired.

Acknowledgments

Thanks to the Crypto 99 program committee for their comments.

The first author was supported in part by a NSF Graduate Fellowship. The second author was supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

References

- 1. J. AN AND M. BELLARE, "Constructing VIL-MACs from FIL-MACs: Message authentication under weakend assumptions," Full version of this paper, available via http://www-cse.ucsd.edu/users/mihir.
- ANSI X9.9, "American National Standard for Financial Institution Message Authentication (Wholesale)," American Bankers Association, 1981. Revised 1986.
- 3. M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
- 4. M. Bellare, R. Canetti and H. Krawczyk, "Pseudorandom functions revisited: the cascade construction and its concrete security," Proceedings of the 37th Symposium on Foundations of Computer Science, IEEE, 1996.
- M. Bellare, A. Desai, E. Jokipii and P. Rogaway, "A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation," Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE, 1997.
- M. Bellare, J. Kilian and P. Rogaway, "The security of cipher block chaining," Advances in Cryptology Crypto 94 Proceedings, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.
- M. Bellare and P. Rogaway, "Collision-Resistant Hashing: Towards Making UOWHFs Practical," Advances in Cryptology - Crypto 97 Proceedings, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed., Springer-Verlag, 1997.
- 8. M. Bellare, O. Goldreich and S. Goldwasser, "Incremental cryptography with application to virus protection," Proc. 27th Annual Symposium on the Theory of Computing, ACM, 1995.
- M. Bellare, R. Guérin and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions," Advances in Cryptology – Crypto 95 Proceedings, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
- 10. L. Carter and M. Wegman, "Universal Hash Functions," *Journal of Computer and System Science*, Vol. 18, 1979, pp. 143–154.
- I. Damgård, "A Design Principle for Hash Functions," Advances in Cryptology Crypto 89 Proceedings, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
- 12. O. GOLDREICH, S. GOLDWASSER AND S. MICALI, "How to construct random functions," *Journal of the ACM*, Vol. 33, No. 4, 210–217, (1986).
- 13. S. Goldwasser and S. Micali, "Probabilistic encryption," Journal of Computer and System Science, Vol. 28, 1984, pp. 270–299.
- S. Goldwasser, S. Micali and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal of Computing, Vol. 17, No. 2, pp. 281–308, April 1988.

- M. Luby and C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions," SIAM Journal of Computing, Vol. 17, No. 2, pp. 373–386, April 1988.
- 16. R. Merkle, "One way hash functions and DES," Advances in Cryptology Crypto 89 Proceedings, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
- R. Merkle, "A certified digital signature," Advances in Cryptology Crypto 89
 Proceedings, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
- M. NAOR AND M. YUNG, "Universal one-way hash functions and their cryptographic applications," Proceedings of the 21st Annual Symposium on Theory of Computing, ACM, 1989.
- E. PETRANK AND C. RACKOFF, CBC MAC for real time data sources. DIMACS Technical Report 97-26, 1997.
- B. PRENEEL AND P. VAN OORSCHOT, "MD-x MAC and building fast MACs from hash functions," Advances in Cryptology – Crypto 95 Proceedings, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
- 21. R. RIVEST, "The MD5 message-digest algorithm," IETF RFC 1321 (April 1992).
- FIPS 180-1. Secure Hash Standard. Federal Information Processing Standard (FIPS), Publication 180-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., April 1995.
- WEGMAN AND CARTER, "New hash functions and their use in authentication and set equality," Journal of Computer and System Sciences, Vol. 22, 1981, pp. 265– 279.

A Proof of Lemma 3

Let C be an arbitrary collision-finder attacking the security of the function g as a weak collision-resistant function and having resources at most t, q and μ as previously defined. We want to upper bound the weak-collision insecurity of g in terms of insecurity of g as a MAC. To do this, we specify an adversary (forger) A that uses the collision-finder C to attack the security of the function family g as a MAC.

Our algorithm for A is composed of two sub-algorithms: A_1 and A_2 . They get an oracle for $g_k(\cdot)$ so that they can mount a chosen-message attack and eventually output a pair (m, τ) . They run C providing answers to C's queries using the oracle they are given. Each of them is allowed to have resources at most t', q' and μ' as defined before. We denote by q_c the number of queries made by C. The algorithms are specified as follows:

Algorithm
$$A_1^{g_k(\cdot)}$$
For $i = 1, \dots, q_c$ do
$$C \to x_i$$

$$C \leftarrow g_k(x_i)$$

$$C \to (y_1, y_2)$$
Choose $m \overset{R}{\leftarrow} \{1, 2\}$
Let $n = \{1, 2\} - \{m\}$
Return $(y_m, g_k(y_n))$

$$Algorithm $A_2^{g_k(\cdot)}$
Choose $i \overset{R}{\leftarrow} \{1, \dots, q_c\}; \quad j \overset{R}{\leftarrow} \{1, \dots, i-1\}$

$$C \to x_s$$
If $s < i$

$$C \leftarrow g_k(x_s)$$
Return $(x_i, g_k(x_j))$$$

To handle the case where both of the final strings y_1 and y_2 that C outputs were queried by the collision-finder C, we have the forger A_2 . For the other case, where at least one of the strings that C outputs was not queried by C, we have the forger A_1 . A_2 makes two random guesses $(i, j, 1 \le j < i \le q_c)$ for the positions in which the queries for the output strings y_1, y_2 are made by C among its query positions $\{1, \ldots, q_c\}$. Without making the query for the guessed second position i, A_2 outputs the forgery using the unqueried string x_i as the new message and the output of the queried string $g_k(x_j)$ as its tag. A_1 randomly chooses one of the output strings of C and outputs that string as the message and uses the other string to obtain its tag.

We now upper bound $\mathbf{Succ}_g^{\mathrm{wcr}}(C)$ in terms of $\mathbf{Succ}_g^{\mathrm{mac}}(A_1)$ and $\mathbf{Succ}_g^{\mathrm{mac}}(A_2)$. For notational convenience, we define the following. Let "C Succeeds" denote the event where the experiment FindWeakCol(C,g) outputs 1. Let $\mathrm{Pr}[\cdot]$ denote the probability of some event occurring in the experiment FindWeakCol(C,g). And let E denote the event where, in the two output strings of C, at least one of them is unqueried.

Forge (A_1,g) will output 1 when FindWeakCol(C,g) outputs 1 (the event "C Succeeds") and at least one of the two output strings of C is unqueried (the event E) and A_1 chooses the unqueried string correctly. Since A_1 chooses the string randomly (out of the two strings), the probability of choosing right is at least 1/2. This means that $\mathbf{Succ}_g^{\mathrm{mac}}(A_1) \geq \frac{1}{2}\Pr[C\ Succeeds \wedge E]$. For the algorithm A_2 , its success requires that it guesses the correct positions for the two queried strings in addition to the success of C with both of its output strings queried. Since the two positions are randomly chosen among q_c numbers, the probability of choosing the correct position pair among the $\binom{q_c}{2}$ possible position pairs is at least $1/\binom{q_c}{2}$. Notice that the event where both of the output strings of C are queried is \bar{E} (from the definition of the event E). Hence, $\mathbf{Succ}_g^{\mathrm{mac}}(A_2) \geq \Pr[C\ Succeeds \wedge \bar{E}]/\binom{q_c}{2}$. Putting all this together we have

$$\mathbf{Succ}_{g}^{\mathrm{wcr}}(C) = \Pr[C \ Succeeds]$$

$$= \Pr[C \ Succeeds \land E] + \Pr[C \ Succeeds \land \bar{E}]$$

$$\leq 2\mathbf{Succ}_{g}^{\mathrm{mac}}(A_{1}) + \binom{q_{c}}{2} \cdot \mathbf{Succ}_{g}^{\mathrm{mac}}(A_{2})$$

$$\leq \frac{q_{c}^{2} - q_{c} + 4}{2} \cdot \mathbf{InSec}_{g}^{\mathrm{mac}}(t', q', \mu') . \tag{5}$$

The analysis of the resource parameters is omitted here, but for completeness is provided in [1].

Regarding Equation 5, the multiplicative factor $\frac{q_c^2 - q_c + 2}{2}$ is less than q^2 since we know that $q_c \leq q$. Combining all this, we obtain the conclusion of Lemma 3.

Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier

Mihir Bellare¹, Oded Goldreich², and Hugo Krawczyk³

Abstract. Many cryptographic solutions based on pseudorandom functions (for common problems like encryption, message-authentication or challenge-response protocols) have the following feature: There is a stateful (counter based) version of the scheme that has high security, but if, to avoid the use of state, we substitute a random value for the counter, the security of the scheme drops below the birthday bound. In some situations the use of counters or other forms of state is impractical or unsafe. Can we get security beyond the birthday bound without using counters?

This paper presents a paradigm for strengthening pseudorandom function usages to this end, the idea of which is roughly to use the XOR of the values of a pseudorandom function on a small number of distinct random points in place of its value on a single point. We establish two general security properties of our construction, "pseudorandomness" and "integrity", with security beyond the birthday bound. These can be applied to derive encryption schemes, and MAC schemes (based on universal hash functions), that have security well beyond the birthday bound, without the use of state and at moderate computational cost.

1 Introduction

Pseudorandom functions [7] are an essential tool in many cryptographic solutions. They can be used to generate a pseudorandom pad for symmetric encryption, to mask a universal hash function for producing a secure message-authentication (MAC), to implement secure challenge-response mechanisms, and so on. In practice, one might use, in the role of pseudorandom functions, various concrete primitives, such as block ciphers or keyed hash functions under the assumption that they do possess the pseudorandomness properties in question.

THE DANGER OF REPETITION. In usages of pseudorandom functions such as those mentioned above, the same pseudorandom function will be applied to many values in the function's domain. In many such cases, security can be compromised if one applies the pseudorandom function twice to the same point.

¹ Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: mihir@cs.ucsd.edu.

URL: http://www-cse.ucsd.edu/users/mihir.

² Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel. E-Mail: oded@wisdom.weizmann.ac.il.

Department of Electrical Engineering, Technion, Haifa 32000, Israel, and IBM T.J. Watson Research Center, New York, USA. Email: hugo@ee.technion.ac.il.

Consider as an example the following method of encryption. Two parties share a key which specifies a function $f: \{0,1\}^n \to \{0,1\}^m$ from some fixed pseudorandom function family. In order to encrypt a message M of length m, the sender computes f on an element $v \in \{0,1\}^n$ and then sends the pair $(v, M \oplus f(v))$. Clearly, the security of such a scheme depends on never re-using the same value v for encrypting different messages. The same problem arises in other applications of pseudorandom functions, including MACs and challenge-response protocols.

USING COUNTERS. A natural way to avoid repetition is for the sender to use (as the points on which to evaluate the function) an increasing counter, or other form of varying, non-repeating state, which is updated with each application of the function. This does very well in terms of avoiding repetition, but can have various drawbacks depending on the setting and application.

Maintaining a counter, or other state information, might in some settings be impractical or unsafe. This can happen, for example, whenever maintaining a synchronized state across different applications of the function is unsafe or impossible. Such is the case of a function that is used across different sessions (or invocations) of a protocol, or used (possibly simultaneously) by different users or components of a system. Additional examples include the use of smartcards, or authentication tokens, that store the key to a pseudorandom function in persistent memory but are not equipped with non-volatile writeable memory to store the varying value of a counter. Even in cases where such a varying state can be stored, security is susceptible to system failures that may reset the value of that counter.

Also some applications require more for security than mere non-repetitiveness of the value to which the pseudorandom function is applied; e.g., the value might be a challenge which should be unpredictable, and a counter value is of course highly predictable. In this case too, the use of counters is not possible at all.

USING COINS. Another possibility is to use $random\ values$ as those on which to evaluate the function. This can avoid the need to store varying information, and also yield unpredictability, thereby avoiding the drawbacks of counters. However, randomness might do less well at the task we first highlighted, namely avoiding repetition. This is due to the "birthday" phenomenon, which means that if the domain of the function has size $N=2^n$, and we apply the function to a sequence of q points selected at random from the domain, we have probability about q^2/N of seeing a repetition in the selected points. In the encryption example discussed above, this represents a significant decrease in the number of messages that can be safely encrypted: only \sqrt{N} if we use random values for the point v, but up to N (depending on the security of the pseudorandom function family) if we use counters.

Thus the birthday bound for query collisions may become the security bottleneck of the whole application. This is particularly evident when using 64-bit input pseudorandom functions, such as those based on DES. In this case a number $q=2^{32}$ of queries nullifies the quantified security; even $q=2^{25}$ leaves us with an insecurity (ie. chance that the scheme may be broken) of $q^2/N=2^{-14}$, which is fairly high. Even with 128-bit blocks (such as in the AES proposals)

	Construction	Insecu	No. f -appls.	
		Upper bound	Lower bound	
1.	CBC-2	$\frac{12q^2}{N}$ [5]	$\Omega(\frac{q^2}{N})$ [14]	2
2.	Feistel- t $(t = 3, 4)$	$\frac{q^2}{N}$ [10]	$\Omega(rac{q^2}{N})$	t
3.	[11]	$O(\frac{q^2}{N})$ [11]	$\Omega(rac{q^2}{N})$	2
4.	Benes [1]	$O\left(\frac{q}{N}\right)$ [1]	$\Omega\left(\frac{q}{N}\right)$	8
5.	$\Omega_t \ (t \ge 1) \ [12]$	$\frac{q^{t+1}}{(t+1)N^t} [12]$?	2t
6.	Feistel-6	$O(\frac{q^4}{N^3} + \frac{q^2}{N^2})$ [13]	?	6

Fig. 1. Input-length doubling transformations: Constructing $g \colon \{0,1\}^{2n} \to \{0,1\}^n$ given $f \colon \{0,1\}^n \to \{0,1\}^n$. The insecurity is the maximum adversarial success in q queries. Both upper bounds and lower bounds (attacks) on the insecurity are shown. Here $N=2^n$. "No. f-apps" is the number of applications of f used in one computation of g, and is the main cost. "Feistel-t" means t rounds, and "CBC-2" means CBC on two blocks. Constructions $\mathbf{2,3,4,6}$ yield maps of 2n bits to 2n bits; in our context it is implicit that the outputs are truncated. Question marks mean we don't know. See the text for (even) more discussion.

the probability of repeated queries leaves less security than usually intended. In this case $q = 2^{32}$ provides 2^{-64} security which is much weaker than the usually conjectured "128-bit security" for these ciphers.

BEATING THE BIRTHDAY BOUND. The above discussion raises the natural question of to what extent the use of varying state (e.g. counters) is *essential* for avoiding the quadratic degradation in the security of the function. In other words, can we combine the advantages of coins and counters: get security beyond the birthday bound, yet avoid the need to maintain state?

USING INPUT-LENGTH DOUBLING TRANSFORMATIONS. One approach is to change the pseudorandom function and use instead one with a larger domain. For example, instead of $f: \{0,1\}^n \to \{0,1\}^m$, we use a pseudorandom function $g: \{0,1\}^{2n} \to \{0,1\}^m$. This however can be impractical, or may not increase security in the desired way, as we now discuss.

Since total redesign of the function is typically not desirable, one would usually try to build g in a generic way from f. Figure 1 summarizes the main known designs. (It sets m=n for simplicity.) For example, one can use the popular CBC-MAC construction. Another alternative is to use one of many known transformations of pseudorandom functions on n bits to pseudorandom permutations (or functions) on 2n bits, and simply drop all but the last m bits of the output. (Constructions 2,3,4,6 of the table fall in this class, while

construction **5** is directly of 2n bits to n bits.) Figure 1 indicates the best known analyses upper bounding the insecurity, the best known attacks lower bounding the insecurity, and the cost measured in terms of the number of applications of f needed to make one computation of g. As the table indicates, the most efficient known constructions are still vulnerable to attacks that in q queries achieve success related to q^2/N where $N=2^n$ is the domain size of the original function. (In particular 1,2,3). The last three constructions have better bounds on the insecurity, but as the table shows, their computational cost (the number of f-applications) is relatively high. In particular, as we will see (Figure 2), it is higher than the cost of our methods discussed below.

Construction. In this paper we propose and investigate a simple mechanism to go beyond the birthday barrier without using counters or state information. We call it the "parity method". Instead of computing the function at a single random point, compute it at several random (but distinct) points (typically two or three points will suffice) and take the parity of the results (namely, XOR these values). For instance, in the above encryption example, if the sender wants to encrypt plaintext M, he will choose two different random values r_1, r_2 from the domain of the function, and send to the other party as the ciphertext the triple $(r_1, r_2, M \oplus f(r_1) \oplus f(r_2))$. Similar methods will be used for other applications such as challenge-response, message authentication, or key derivation. As a result our methods offer a sateless alternative to achieve the high security of stateful schemes at a moderate computational cost but with increased use of random bits.

SECURITY. We are interested in proving general security properties of the parity method that can later be applied to prove the security of specific encryption schemes (such as the one discussed above) or MAC schemes (such as we will discuss below). Accordingly, we begin by considering the probabilistic function that embodies the parity construct, namely

$$F(r_1, \dots, r_t) = \bigoplus_{i=1}^t f(r_i) \tag{1}$$

where the r_i 's are uniformly chosen different n-bit numbers. The first security property we consider is pseudorandomness, or "distinguishability distance" from true randomness, of the (randomized) function F. This corresponds to passive attacks. The second security property we call "integrity", and it corresponds to certain kinds of active attacks. (In the coming sections we will discuss these properties in more depth, and see how they apply to encryption and MAC respectively.) In either case we are interested in how the security of this randomized function degrades after q queries relative to the security of the original pseudorandom function f. Our analyses reduce this question to a purely information-theoretic setting, and show that the parity method amplifies security at quite a high rate, enabling one to move well beyond the birthday barrier. Our results are displayed in Figure 2 and discussed below.

PSEUDORANDOMNESS AMPLIFICATION AND ENCRYPTION. An adversary sees q vectors (r_1, \ldots, r_t) and the output of the parity function on them. We define a certain "bad" event and show that subject to its not happening, the outputs

	Property	Insecu	No. f -appls.	
		Upper bound	Lower bound	
1.	Pseudorandomness	$O(t!) \cdot rac{q^2}{N^t}$	$arOmega(t!)\cdotrac{q^2}{N^t}$	t
2.	Integrity	$(t \lg N)^{O(t)} \cdot \frac{q^3}{N^t}$	$arOmega(t^t)\cdotrac{q^3}{N^t}$	t

Fig. 2. The two security properties of the t-fold parity construction for $t \ge 1$: Parameters are as in Figure 1. This is true for q < N/O(t), and t is odd in 2.. Bounds shown are approximate.

look uniform. Exploiting and extending a connection of [4], the bad event is that a certain matrix associated to the vectors is not of full rank. Lemma 2 bounds this probability roughly by:

$$d_1(t) \cdot \frac{q^2}{N^t}$$
 for $q \le \frac{N}{e^2 t}$ and $d_1(t) = 0.76 \cdot t!$, (2)

where $N=2^n$ is the size of the domain of the function. (The bound on q is necessary: given the q sequences of $(r_1,...,r_t)$'s, the randomness in the process of Equation (1) is only due to f itself which has N bits of randomness.) Remarkably, the bound Equation (2) shows that if f is chosen as a truly random function then the effect of the parity construct of Equation (1) on limiting the degradation of security due to repeated queries is, for q < O(N/t) and small t, close to the effect of applying a random function on single inputs of length tn. Indeed, in the latter case the distance from randomness is, using the birthday argument, of the order of $\frac{q^2}{N^t}$. That is, we approximate the effect of a t-fold increase in the queries size without necessitating any change to the underlying function f. We note that the bound is tight.

The encryption scheme discussed above, a special case of the CTR scheme in [2], was shown by the latter to have insecurity (under a chosen-plaintext attack of q < N messages) at most ϵ , the maximum possible attainable advantage in breaking the underlying pseudorandom function in q queries and time related to that allowed the encryption attacker. The insecurity of the randomized (stateless) version is only bounded by $\epsilon + q^2/N$ due to birthday attacks. In Section 3 we consider the (counter-less) encryption scheme in which to encrypt plaintext M, we choose t distinct random values r_1, \ldots, r_t and set the ciphertext to $(r_1, \ldots, r_t, F(r_1, \ldots, r_t) \oplus M)$. Theorem 1 bounds its insecurity by the term of Equation (2) modulo an additive term corresponding to the insecurity of F under tq queries. Considering the case t=2 discussed above, for $q=O(\sqrt{N})$, the new scheme has security which is close to the counter-version of the basic CTR scheme, whereas the coin-version of the basic scheme is totally insecure at $q=\sqrt{N}$. Furthermore the security gets even better with larger t.

INTEGRITY AMPLIFICATION AND MESSAGE AUTHENTICATION. In the Carter-Wegman paradigm [16], the MAC of message M is $(C, h(M) \oplus f(C))$, where C is a counter value, f is a pseudorandom function (PRF), and h is a δ -AXU hash function [9]. When trying to make this stateless by substituting a random string for C, security drops to the birthday bound. The same situation arises in the XOR MAC schemes of [4]. A counter based variant of their scheme has high security, but the stateless version substitutes a random value for the counter and security drops to the birthday bound. The modified (stateless) Carter-Wegman MAC scheme we propose is that the MAC of message M be $(r_1, \ldots, r_t, h(M) \oplus F(r_1, \ldots, r_t))$ where $r_1, \ldots, r_t \in \{0, 1\}^n$ are random but distinct points, and f, h are as before. Here f is a parameter, and the higher we set it, the more security we get, though each increment to f costs one extra application of the PRF.

The pseudorandomness of the parity construct does not by itself guarantee security of the above due to the fact that an adversary in a MAC setting is allowed an active attack, and can attempt a forgery in which the values r_1, \ldots, r_t are of its own choice. We propose another property of the parity construct we call "integrity". We again reduce the analysis to the question of whether the matrix associated to the points on which the parity function is evaluated has a certain property, which we call "vulnerability" and is defined in Section 4. Improvement over the birthday bound occurs only at $t \geq 3$. Specifically, for odd t, Lemma 4 bounds the probability of vulnerability by

$$d'(t, \lg N) \cdot \frac{q^3}{N^t}$$
 for $q \le \frac{N}{2e^2t}$, (3)

where $N=2^n$ and $d'(t,\lg N)$ is a polynomial in $\lg N$ for each fixed t, whose value is specified by Equation (13). (Curiously enough, the bound for even $t \geq 4$ is typically inferior to the bound for t-1. Specifically, for even t our bound is $d'(t,\lg N) \cdot \frac{q^2}{N^{t/2}}$, which is tight.) Note that this expression is inferior to the one obtained in Equation (2). Still, it suffices for our applications. We apply this to get Theorem 2, an analysis of the security of the MAC scheme discussed above.

DISCUSSION AND RELATED WORK. One should note that getting security beyond the birthday bound (both in the case where one uses counters, and in our setting where one does not) requires that we use a pseudorandom function family which itself has security beyond the birthday bound. This precludes the direct use of block ciphers; since they are permutations, their security does not go beyond the birthday bound. The question of designing pseudorandom functions (with security beyond the birthday bound) out of pseudorandom permutations (which model block ciphers) was first considered by Bellare, Krovetz and Rogaway [6] and later by Hall, Wagner, Kelsey and Schneier [8]. These two works provide several constructions that one might use. The works of [6, 8] were also motivated by the desire to get beyond the birthday bound for encryption, but were using a counter-based encryption scheme: their applications are not stateless.

Shoup [15] considers various ways of providing better security tradeoffs when using pseudorandom functions or permutations as masks in universal-hash function based MACs. He gets the security to decrease slower as a function of the

number of queries, but does not get security beyond the birthday bound without the use of state.

2 Definitions

Primitives discussed in this paper include pseudorandom function families [7], symmetric encryption schemes, and MACs. Security of all these will be treated in a concrete framework along the lines of works like [5,2]. Since this approach is by now used in many places, we will briefly summarize the concepts and terms we need.

The definitional paradigm we employ is to associate to any scheme an *insecu*rity function which, given some set of parameters defining resource limitations, returns the maximum possible success probability of an adversary limited to the given resources. The definition of "success" various with the goal of the primitive, as do the resources considered. The following will suffice either for an experienced reader or for one wanting to understand our results at a first, high level. More precise definitions can be found in [3].

PSEUDORANDOM FUNCTION FAMILIES. [Notion of [7], concretized as per [5]]. To a family F of functions (in which each function maps $\{0,1\}^n$ to $\{0,1\}^m$) we associate an *insecurity function* $\mathbf{InSec}^{\mathrm{prf}}(F,\cdot,\cdot)$ defined as follows: For integers q,T the quantity $\mathbf{InSec}^{\mathrm{prf}}(F,q,T)$ is the maximum possible "advantage" that an adversary can obtain in distinguishing between the cases where its given oracle is a random member of F or a truly random function of $\{0,1\}^n$ to $\{0,1\}^m$, when the adversary is restricted to q oracle queries and running time T.

SYMMETRIC ENCRYPTION SCHEMES. [Following [2]]. To a symmetric encryption scheme ENC (consisting of a probabilistic encryption algorithm and deterministic decryption algorithm) we associate an insecurity function $\mathbf{InSec}^{\mathrm{enc}}(\mathsf{ENC},\cdot,\cdot)$ defined as follows: For integers μ,T the quantity $\mathbf{InSec}^{\mathrm{enc}}(\mathsf{ENC},\mu,T)$ is the maximum possible probability that an adversary can "break" the encryption scheme under a chosen-plaintext attack in which a total of μ plaintext bits are encrypted and the running time of the adversary is restricted to T. ("Break" here means in the sense of real-or-random security [2] .)

MACs. [Following [4]]. To a message authentication scheme MAC (consisting of a probabilistic mac generation algorithm and deterministic mac verification algorithm¹) we associate an insecurity function $\mathbf{InSec}^{\mathrm{mac}}(\mathsf{MAC}, \cdot, \cdot, \cdot)$ defined as follows: For integers q_a, q_v, T the quantity $\mathbf{InSec}^{\mathrm{mac}}(\mathsf{MAC}, q_a, q_v, T)$ is the maximum possible probability that an adversary can forge a mac of a new message under an attack in which it obtains valid macs of q_a texts of its choice, verifies up to q_v candidate message/mac pairs of its choice, and runs in time at most T.

Conventions. In any insecurity function, we might drop the time argument T, and it is to be understood then that the time allowed the adversary is not

¹ Traditional MACs are deterministic, so verification can be done by mac recomputation. Our mac generation process is probabilistic, so a separate verification procedure must be prescribed.

restricted, meaning we are in an information theoretic setting. Indeed, this will be the important case in analyses.

3 Pseudorandomness of Parity

We need a bit of terminology. A sequence $R = (r_1, \ldots, r_t)$ of n-bit strings is called non-colliding if the t strings r_1, \ldots, r_t are all distinct. We let D(n, t) denote the set of all non-colliding t-sequences of n-bit strings. We let R(n, m) denote the set of all functions of $\{0, 1\}^n$ to $\{0, 1\}^m$.

Parity distribution. Consider the following game. A random function f from R(n,m) is chosen and fixed. Then q non-colliding sequences, $R_i = (r_{i,1}, \ldots, r_{i,t})$ for $i=1,\ldots,q$, are chosen randomly and independently. An adversary is provided these sequences together with the q corresponding output values of the parity function, namely $b_i = f(r_{i,1}) \oplus \cdots \oplus f(r_{i,t})$ for $i=1,\ldots,q$. In applications, it is typical that as long as b_1,\ldots,b_q look like random independent m-bit strings (given the other information), the adversary will not be able to derive any "advantage" in "breaking" the security of the application, whatever that may be. This will be seen more clearly and specifically later, but for the moment we wish only to give some clue as to the motivation for what we now look at. Namely, the experiment which produces the output just described, which we call Par(n,q,t). We wish to "compare" this to the output of the experiment which picks R_1,\ldots,R_q the same way, and b_1,\ldots,b_q randomly. The experiments are described below.

Experiment
$$\operatorname{Par}(n,q,t)$$
 $f \overset{R}{\leftarrow} R(n,m)$
For $i=1,\ldots,q$ do
$$R_i = (r_{i,1},\ldots,r_{i,t}) \overset{R}{\leftarrow} D(n,t)$$
 $b_i \leftarrow \bigoplus_{j=1}^t f(r_{i,j})$
End do
Output (R_1,b_1,\ldots,R_q,b_q)
Experiment $\operatorname{Rnd}(n,q)$

$$R_i = (r_{i,1},\ldots,r_{i,t}) \overset{R}{\leftarrow} D(n,t)$$
 $b_i \overset{R}{\leftarrow} \{0,1\}^m$
End do
Output (R_1,b_1,\ldots,R_q,b_q)

A natural comparison measure is the statistical distance between the output distributions of these experiments, and we would like to upper bound it. In fact we will need a stronger claim. We will define a certain "bad" event, and upper bound its probability. We will also assert that conditioned on the bad event not occurring, the outputs of the two experiments are identically distributed. (The bad event will depend only on the choices of R_1, \ldots, R_q hence is defined and has the same probability under both experiments.) In other words, when the bad event does not occur, the outputs b_1, \ldots, b_q of the parity experiment are random and uniform. It follows in particular that the statistical distance between the output distributions of the two experiments is bounded by the probability of the bad event, but applications will in fact exploit the stronger assertion.

MATRIX TO PSEUDORANDOMNESS CONNECTION. The definition of the bad event is based on an association of a matrix to the parity distribution. This connection is taken from [4], where it is used to analyze a MAC construction based on the

XOR operation. We adapt it for our purposes. Then the bulk of our analysis focuses on this matrix. Let us now describe the matrix and explain more precisely the connection to the pseudorandomness of parity.

To any non-colliding sequence $R=(r_1,\ldots,r_t)$ of n-bit strings is associated its characteristic vector of length $N=2^n$, denoted $\operatorname{ChVec}(R)$. Namely, if we consider the values r_i as representing integer numbers between 0 and N-1 then the characteristic vector of r_1,\ldots,r_t will have a value of 1 in the positions corresponding to these t numbers and 0 elsewhere. If R_1,\ldots,R_q are non-colliding sequences we denote by $\operatorname{MTX}_{N,q}(R_1,\ldots,R_q)$ the q by N matrix (of zeros and ones) whose i-th row is $\operatorname{ChVec}(R_i)$ for $i=1,\ldots,q$. We are interested in the rank of our matrix when it is viewed as a random variable over the choices of R_1,\ldots,R_q from D(n,t). This is captured by the following quantity:

$$\begin{aligned} &\mathsf{NFRProb}(N,q,t) \\ &= \Pr \left[\, \mathsf{MTX}_{N,q}(R_1,\ldots,R_q) \,\, \mathrm{is \,\, not \,\, of \,\, full \,\, rank} \,: \,\, R_1,\ldots,R_q \, \stackrel{R}{\leftarrow} D(n,t) \, \right] \,. \end{aligned}$$

Now, let $b_i = f(r_{i,1}) \oplus \cdots \oplus f(r_{i,t})$ for $i = 1, \ldots, q$. View the values b_1, \ldots, b_q as arranged in a column vector consisting of q strings, each m-bits long. Then notice that this vector is given by the following matrix vector product, where as before we identify $\{0,1\}^n$ with $\{0,1,\ldots,N-1\}$ for simplicity:

$$MTX_{N,q}(R_1, \dots, R_q) \cdot \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{bmatrix} . \tag{4}$$

Namely $b_1 = f(r_{1,1}) \oplus \cdots \oplus f(r_{1,t}) = \sum_j f(j)$, the sum being taken over all values j for which the j-th coordinate of $\operatorname{ChVec}(R_1)$ is 1, and so on.

The following lemma says that as long as the matrix has full rank, the entries of the output vector are uniformly and independently distributed over $\{0,1\}^m$. That is, they look like the outputs of a random function with range $\{0,1\}^m$ being evaluated at q distinct points. It is an adaption of a lemma of [4] to our setting, and is informally stated.

Lemma 1. Conditioned on the event that $MTX_{N,q}(R_1, ..., R_q)$ is of full rank, the outputs of experiment Par(n,q,t) and experiment Rnd(q,t) are identically distributed.

The implication in terms of the usage of the parity construct is that as long as the matrix maintains full rank, seeing the outputs of the parity construct yields no information at all to an adversary. It is just like seeing values of a random function on distinct points. Accordingly, adversarial success will only happen when the matrix is *not* of full rank. For this reason, our efforts are concentrated on upper bounding NFRProb(N,q,t).

The heart of our analysis reduces by the above to upper bounding the probability that the matrix $\mathrm{MTX}_{N,q}(R_1,\ldots,R_q)$ is not of full rank when R_1,\ldots,R_q are randomly and independently chosen non-colliding vectors. The bound is given

in terms of $N=2^n, t$ and q in the following lemma. Here e is the base of the natural logarithm.

Lemma 2. Let t be such that $1 \le t \le \sqrt{N/(e \lg N)}$, then for any $q < N/(e^2t)$ we have

$$\mathsf{NFRProb}(N,q,t) \ \leq \ d_1(t) \cdot \frac{q^2}{N^t} \ + \ \begin{cases} d_2(t,\lg N) \cdot \frac{q^3}{N^{3t/2}} \ \textit{if t is even} \\ \\ d_2(t,\lg N) \cdot \frac{q^4}{N^{2t}} \ \textit{if t is odd} \ , \end{cases} \tag{5}$$

where $d_1(t) = 0.76 \cdot t!$ and

$$d_2(t,n) = \begin{cases} 3e^{3+3t/2}2^{-3}t^{-3+3t}n^{-3+3t/2} & \text{if } t \text{ is even} \\ e^{4+2t}2^{-4}t^{-4+4t}n^{-4+2t} & \text{if } t \text{ is odd.} \end{cases}$$
(6)

DISCUSSION OF THE BOUNDS. Let us now interpret the bounds a bit. First, the upper bound on t is a technicality insignificant in practice, and safely ignored. (For example if $N=2^{64}$ it says roughly that $t\leq 2^{29}$, and we are interested in values like t=2,3,4,5.) The bound on q indicates that we are not expecting security for q above N; in fact q must be O(N). This is necessary, as noted in the introduction, for entropy reasons alone. The main thing is Equation (5) which says that NFRProb(N,q,t) is roughly bounded by q^2/N^t . This is modulo a small constant factor, and also an additive term. The additive term has a factor of $q^s/N^{st/2}$ with $s\geq 3$, which is small enough to make the whole additive term negligible, even given the somewhat large seeming coefficient $d_2(t, \lg N)$. Accordingly it is safe to view the above bound as essentially $d_1(t) \cdot q^2/N^t$.

Example 1. Take for example $N=2^{64}$ and t=3. Then $d_1(t)\leq 4.6$ and $d_2(3,64)<2^{31}$ so

$$\mathsf{NFRProb}(N,q,3) \ \leq \ 4.6 \cdot \frac{q^2}{N^3} \ + \ 2^{31} \cdot \frac{q^4}{N^6} \ \leq \ 4.6 \cdot \frac{q^2}{2^{64*3}} \ + \ \frac{q^4}{2^{64*6-31}} \ \leq \ 5 \cdot \frac{q^2}{N^3} \ .$$

as long as $q \leq N/23$. Thus, we are off from q^2/N^3 by only the small factor of 5. Note in particular the bound is essentially equal to $d_1(t) \cdot q^2/N^t$.

TIGHTNESS OF THE ABOVE BOUND. The above upper bound can be proven to be approximately tight by considering the event in which two rows in $\mathrm{MTX}_{N,q}(R_1,\ldots,R_q)$ are identical. This is an instance of the usual birthday paradox: We are selecting q rows from a universe of $\binom{N}{t}$ possible rows. Then a standard birthday calculation (we take the specific estimates used here from [4]) says that for $2 \leq q \leq \sqrt{\binom{N}{t}}$ the probability of collisions is at least

$$0.16 \cdot \frac{q^2}{\binom{N}{t}} \; \geq \; 0.16 \cdot \frac{q^2}{N^t/t!} \; \geq \; 0.16 \cdot t! \cdot \frac{q^2}{N^t} \; .$$

Comparing with the first term in the bound of Lemma 2 we see that the bounds are tight to within a constant that is independent of N, t, q.

280

Proof of Lemma 2: The case of t=1 corresponds to the well-known birthday bound (i.e., we are interested in the probability that two rows have their single 1-entry in the same column). The proof thus focuses on (and assumes) $t \geq 2$. In the following, it is understood that the probabilities are over the choices of R_1, \ldots, R_q uniformly and independently from D(n, t).

 $\begin{aligned} & \mathsf{NFRProb}(N,q,t) \\ &= \sum_{i=2}^{q-1} \Pr[\mathsf{MTX}_{N,q}(R_1,\ldots,R_q) \text{ has rank } i] \\ &\leq \sum_{i=2}^{q-1} \sum_{1 \leq j_1 < \cdots < j_i \leq q} \Pr[\mathsf{Rows}\ j_1,\ldots,j_i \text{ of } \mathsf{MTX}_{N,q}(R_1,\ldots,R_q) \text{ sum to zero}] \,. \end{aligned}$

Let p(N, i, t) denote the probability that a *i*-by-N matrix over Z_2 , in which each row is a random N-string with exactly t ones, has row-sum zero. Since the probability above does not depend on which rows we consider we have

$$\mathsf{NFRProb}(N,q,t) \ \leq \ \sum_{i=2}^{q-1} \binom{q}{i} \cdot p(N,i,t) \ .$$

Notice that if t is odd then three rows of the matrix cannot sum to zero. So set s = 3 if t is even and s = 4 if t is odd. Then our bound becomes

$$\mathsf{NFRProb}(N,q,t) \leq \binom{q}{2} \cdot p(N,2,t) + \sum_{i=s}^{q-1} \binom{q}{i} \cdot p(N,i,t) \ . \tag{7}$$

Claim: For any $2 \le i \le q-1$ we have

$$p(N, i, t) \le \begin{cases} \frac{2d_1(t)}{N^t} & \text{if } i = 2\\ \left(\frac{eti}{2N}\right)^{ti/2} & \text{if } i \ge 3 \end{cases}.$$

Proof of Claim: Let R denote a matrix selected according to the above distribution. If i=2 then p(N,2,t) is just the probability of a collision when two balls are thrown into $\binom{N}{t}$ buckets. This is

$$\frac{1}{\binom{N}{t}} = \frac{t!(N-t)!}{N!} = \frac{t!}{N(N-1)\cdots(N-t+1)} \le \frac{t!}{(N-t+1)^t}.$$

By assumption $t \leq \sqrt{N/(e \lg N)}$ so we can lower bound the numerator by

$$\left(N-\sqrt{N}\right)^t \ = \ N^t \cdot \left(1-\frac{1}{\sqrt{N}}\right)^t \ \geq \ N^t \cdot \left(1-\frac{t}{\sqrt{N}}\right) \ \geq \ N^t \cdot \left(1-\frac{1}{(e\lg N)^{1/2}}\right) \ .$$

The lowest value of N meeting the conditions in the lemma statement is N=9 and hence the above is at most $0.659 \cdot N^t$. Putting all this together we get

$$p(N,2,t) \ \leq \ \frac{1.517 \cdot t!}{N^t} \ \leq \ 2d_1(t) \cdot N^{-t}$$

as desired.

Now consider $i \geq 3$. Each column in R having some 1-entry, must have at least 2 such entries. Thus, the probability that the rows of R sum to zero is upper bounded by the probability that R has 1-entries in at most it/2 columns. We can view the choice of a row as that of picking at random a subset of exactly t columns in which to place ones. Thus

$$p(N,i,t) \ \leq \ \binom{N}{ti/2} \cdot \left\lceil \frac{\binom{ti/2}{t}}{\binom{N}{t}} \right\rceil^i \ = \ \binom{N}{ti/2} \cdot \left\lceil \frac{\prod_{j=0}^{t-1} \frac{ti}{2} - j}{\prod_{j=0}^{t-1} N - j} \right\rceil^i \ .$$

Now use the fact that $a \leq b$ implies $(a-1)/(b-1) \leq a/b$. This can be applied since $ti/2 \leq N/2$, the latter being true because $i \leq q \leq N/(2e^2t)$. This bounds the above by

$$\binom{N}{ti/2} \cdot (ti/2N)^{ti} \leq \left(\frac{Ne}{ti/2}\right)^{ti/2} \cdot (ti/2N)^{ti}.$$

Simplifying the last term yields the claim. \Box

From Equation (7) and the Claim we get

$$\begin{aligned} \mathsf{NFRProb}(N,q,t) &\leq \binom{q}{2} \cdot p(N,2,t) \; + \; \sum_{i=s}^{q-1} \left(\frac{qe}{i}\right)^i \cdot \left(\frac{eti}{2N}\right)^{ti/2} \\ &= \binom{q}{2} \cdot \frac{2d_1(t)}{N^t} \; + \; \sum_{i=s}^{q-1} \left[eq \cdot \left(\frac{et}{2N}\right)^{t/2} \cdot i^{\frac{t}{2}-1}\right]^i \; . \end{aligned} \tag{8}$$

The first term of Equation (8) is at most $q^2/2 \cdot 2d_1(t)/N^t = d_1(t) \cdot q^2/N^t$. This yields the first term in the bound claimed in the lemma statement. Now we consider the sum

$$S = \sum_{i=s}^{q-1} \left[eq \cdot \left(\frac{et}{2N} \right)^{t/2} \cdot i^{\frac{t}{2}} - 1 \right]^{i}$$

and show it is bounded by the second term in the lemma statement.

Let α be a value to be determined. Then some calculations show that

$$S \le \sum_{i=s}^{\alpha \lg N} \left[eq \cdot \left(\frac{et}{2N} \right)^{t/2} \cdot (\alpha \lg N)^{\frac{t}{2} - 1} \right]^i + \sum_{i=1+\alpha \lg N}^q \left[e \cdot \left(\frac{etq}{2N} \right)^{t/2} \right]^i$$
(9)

We will impose upper bounds on q that guarantee

$$A \stackrel{\text{def}}{=} eq \cdot \left(\frac{et}{2N}\right)^{t/2} \cdot (\alpha \lg N)^{\frac{t}{2} - 1} \le \frac{1}{2} \text{ and } B \stackrel{\text{def}}{=} e \cdot \left(\frac{etq}{2N}\right)^{t/2} \le \frac{1}{2}. \quad (10)$$

In that case, each of the sums of Equation (9) is bounded by twice its first term, so we can bound the sum itself by

$$2 \cdot \left[eq \cdot \left(\frac{et}{2N} \right)^{t/2} \cdot (\alpha \lg N)^{\frac{t}{2}} - 1 \right]^s + \left[e \cdot \left(\frac{etq}{2N} \right)^{t/2} \right]^{\alpha \lg N}$$

$$\leq \left[2e^{\frac{st}{2} + s} (t/2)^{st/2} (\alpha \lg N)^{\frac{st}{2} - s} \right] \cdot \frac{q^s}{N^{st/2}} \ + \ 2^{-\alpha \lg N} \ .$$

Now set $\alpha = 2t$. The second term is $N^{-\alpha} = N^{-2t}$ and hence we get

$$S \,\, \leq \,\, \left[3 e^{\frac{st}{2} + s} t^{st-s} 2^{-s} (\lg N)^{\frac{st}{2} - s} \right] \cdot \frac{q^s}{N^{st/2}} \,\, .$$

To complete the proof, put this together with the above, plug in the appropriate value of s=3 if t is even and s=4 if t is odd, and simplify. This yields the bound in the lemma statement.

It remains to see what conditions on q, t are imposed by Equation (10). Recalling that $\alpha = t$, some calculations show that the conditions imposed by $A \leq 1/2$ and $B \leq 1/2$ are, respectively,

$$q \le \frac{t \lg N}{e} \left(\frac{N}{et^2 \lg N} \right)^{t/2}$$
 and $q \le \frac{N}{e^2 t}$.

As long as $N \ge et^2 \lg N$, some more calculation shows that

$$\frac{N}{e^2 t} \le \frac{t \lg N}{e} \left(\frac{N}{e t^2 \lg N} \right)^{t/2}.$$

To ensure $N \ge et^2 \lg N$ we have made the requirement $t \le \sqrt{N/(e \lg N)}$. Now if $q \le N/e^2t$ then we are ensured $A, B \le 1/2$. The proof is complete.

CTR MODE ENCRYPTION. Let F be a family of functions with domain $\{0,1\}^n$ and range $\{0,1\}^m$. In this section we look at the problem of encrypting a message of m-bits. (In [3] we discuss how to encrypt messages of longer and varying lengths.)

A standard mode to encrypt an m-bit message M is to pick a value $r \in \{0,1\}^n$ and set the ciphertext to $(r,f(r)\oplus M)$. Here $f\in F$ is the (secret) key under which encryption and decryption are performed. The counter version sets r to a counter value that is incremented with each message encrypted. Denoting it by StandardENC-Ctr, the insecurity is shown in [2] be be bounded as indicated below. For any number q < N of m-bit messages queried in a chosen-plaintext attack, setting $N = 2^n$ -

$$\mathbf{InSec}^{\mathrm{enc}}(\mathsf{StandardENC-Ctr}, qm, T) \leq 2 \cdot \mathbf{InSec}^{\mathrm{prf}}(F, q, T') + 2^{-m}$$
. (11)

Here T' = T + O(q(n+m)). When a stateless scheme is desired, the standard paradigm would pick r at random. A chosen-plaintext attack of q messages results in a collision in r values with probability $O(q^2/N)$, and when this happens the encryption scheme is broken, in the sense that partial information about the plaintext is leaked. We wish to apply the parity construct to get better security, comparable or superior to that of the counter version.

OUR SCHEME. The idea is that instead of picking one point r, the encryptor picks t distinct random points r_1, \ldots, r_t , and sets the ciphertext of M to $(r_1, \ldots, r_t, f(r_1) \oplus \cdots \oplus f(r_t) \oplus M)$, the setting being the same as above.

More precisely, we associate to F an encryption scheme $\mathsf{ENCRX}_t[F]$, parameterized by the integer $t \geq 1$. It consists of two algorithms, one to encrypt and

ENCRX_t[F]: encryption procedure INPUT: Key f, plaintext M Pick distinct, random points $r_1, \ldots, r_t \in \{0, 1\}^n$ Let $mk = f(r_1) \oplus f(r_2) \oplus \cdots \oplus f(r_t)$ Let $mdM = mk \oplus M$ Return (r_1, \ldots, r_t, mdM) Return (r_1, \ldots, r_t, mdM)

Fig. 3. ENCRX_t[F]: Our encryption scheme: Here $M \in \{0,1\}^m$ is the plaintext and $f \in F$ is the key.

the other to decrypt. These algorithms are described in Figure 4. The encryption algorithm takes as input a key f and a message $M \in \{0,1\}^m$, while the decryption algorithm takes the same key and a ciphertext. Here f is a random member of F. It is understood that f is accessible as an oracle. (When F is pseudorandom, a seed explicitly supplied to the algorithms names a particular function in the family and thus enables computation of the oracle. But the view of f as an oracle better suits the analysis.)

The security of our scheme can be analyzed via a connection to matrix rank and Lemma 2, as detailed in [4], to yield the following.

Theorem 1. Let F be a family of (pseudorandom) functions with domain $\{0,1\}^n$ and range $\{0,1\}^m$, and let $N=2^n$. Let $t \geq 1$ and let $\mathsf{ENCRX}_t[F]$ be the associated encryption scheme as defined above. Assume $1 \leq q \leq N/(e^2t)$. Then

$$\mathbf{InSec}^{\mathrm{enc}}(\mathsf{ENCRX}_t[F],qm,T) \leq d_1(t) \cdot \frac{q^2}{N^t} + 2 \cdot \mathbf{InSec}^{\mathrm{prf}}(F,tq,T') ,$$
 where $T' = T + O(tq(n+m))$ and $d_1(t)$ is as in Equation (2).

4 Integrity of Parity and Application to MACs

When the parity construct is used in an application such as MAC where the adversary is active, further properties are required to ensure security. It turns out we need to consider the following. An adversary A sees an output $(R_1, b_1, \ldots, R_q, b_q)$ of experiment $\operatorname{Par}(n,q,t)$. Now A tries to create a non-colliding sequence $R_{q+1} = (r_{q+1,1}, \ldots, r_{q+1,t})$ and a value b_{q+1} such that $R_{q+1} \notin \{R_1, \ldots, R_q\}$ and $b_{q+1} = f(r_{q+1,1}) \oplus \cdots \oplus f(r_{q+1,t})$. Notice that this is easy for A to do if there is some subset S of the rows of $\operatorname{MTX}_{N,q}(R_1, \ldots, R_q)$ which sums up to a N-vector v of exactly t ones, because then A can define R_{q+1} via $v = \operatorname{ChVec}(R_{q+1})$ and then set b_{q+1} to $\oplus_i b_i$, the XOR being over all i such that $\operatorname{ChVec}(R_i)$ is a row in S. We will see that in fact this is the only condition under which A can do it. Thus we want to make sure no subset of rows S has this property. This will imply that if A creates some non-colliding sequence $R_{q+1} \notin \{R_1, \ldots, R_q\}$, then A's chance of predicting $f(r_{q+1,1}) \oplus \cdots \oplus f(r_{q+1,t})$ correctly is at most 2^{-m} . Based on this it will be possible to prove the security of our MAC scheme.

The problem can be formulated by extending the experiments $\operatorname{Par}(n,q,t)$ and $\operatorname{Rnd}(n,q)$ to consider an adversary as discussed above. However since we went through that approach before, we will not do it again. Rather we will skip to the essential step and lemma based on which we can directly prove the security of the applications. This lemma is again about the probability that $\operatorname{MTX}_{N,q}(R_1,\ldots,R_q)$ has certain properties.

We need to consider the probability that one may augment the given matrix $MTX_{N,q}(R_1,\ldots,R_q)$ by a row with t 1-entries, different from all current rows, so as to result in a matrix of rank at most q. Actually, we will ask for a little more, to simplify the analysis.

We say a subset S of its rows sums is bad if it sums up to a N-vector v such that $v \not\in S$ but v contains exactly t 1-entries. We say that $MTX_{N,q}(R_1,\ldots,R_q)$ is t-vulnerable if one of the following is true: (1) It has two identical rows, or (2) some subset of its rows is bad. We let

$$\begin{aligned} & \mathsf{VulProb}(N,q,t) \\ &= \Pr \left[\, \mathsf{MTX}_{N,q}(R_1,\ldots,R_q) \,\, \mathrm{is} \,\, t\text{-vulnerable} \,: \, R_1,\ldots,R_q \, \stackrel{R}{\leftarrow} D(n,t) \, \right] \,. \end{aligned}$$

The following lemma considers an arbitrary adversary that given an output of experiment Par(n, q, t) attempts to create a new R_{q+1} and the corresponding f value. It says that A has no better strategy than to guess, as long as the matrix is not t-vulnerable.

Lemma 3. Fix any adversary A that on any input $(R_1, b_1, \ldots, R_q, b_q) \in D(n, t) \times \{0, 1\}^m \times \cdots \times D(n, t) \times \{0, 1\}^m$ outputs some $R_{q+1} = (r_{q+1,1}, \ldots, r_{q+1,t}) \in D(n, t) - \{R_1, \ldots, R_q\}$ and a string $b_{q+1} \in \{0, 1\}^m$. In experiment Par(n, q, t), conditioned on the event that $MTX_{N,q}(R_1, \ldots, R_q)$ is not t-vulnerable, the probability that $b_{q+1} = f(r_{q+1,1}) \oplus \cdots \oplus f(r_{q+1,t})$ is at most 2^{-m} .

Motivated by this we proceed to bound $\mathsf{VulProb}(N,q,t)$ (the proof of next lemma is omitted – see [3]).

Lemma 4. Let t be such that $1 \le t \le \sqrt{N/(2e \lg N)}$, then for any $q < N/(2e^2t)$ we have

$$\mathsf{VulProb}(N,q,t) \leq \begin{cases} d'(t,\lg N) \cdot \frac{q^2}{N^{t/2}} \text{ if } t \text{ is even} \\ \\ d'(t,\lg N) \cdot \frac{q^3}{N^t} \text{ if } t \text{ is odd}, \end{cases}$$
 (12)

where

$$d'(t,n) = \begin{cases} e^{2+3t/2} 2^{3t/2} 3^{-t/2} t^{-2+3t/2} n^{t-2} & \text{if } t \text{ is even} \\ e^{3+2t} 2^{-3} t^{-3+5t/2} n^{t-2} & \text{if } t \text{ is odd.} \end{cases}$$
(13)

Notice the difference in the bounds for odd versus even t. We will focus on odd t. In comparison with Lemma 2 the main term in the bound, namely q^3/N^t , has an extra factor of q. Other than that things are pretty similar. To get an idea of

the relative values of the various terms, consider $N=2^{64}$ and t=3. Then the lemma says that for $q \leq N/46$ we have $\mathsf{VulProb}(N,q,3) \leq 2^{24} \cdot q^3/N^3$.

TIGHTNESS OF THE ABOVE BOUND. Suppose that q < N (which is required and assumed anyhow). Consider, first, an even t. Then the probability that a q-by-N matrix is t-vulnerable is lower bounded by $\Omega(q^2)$ times the probability that two t-vectors add-up to another t-vector. The probability for this event is computed by first selecting and fixing the first vector, and next computing probability that the second vector agrees with it on exactly t/2 1-entries. The latter probability is $\Theta((t/N)^{t/2})$.

For odd t, we consider the event that three distinct t-vectors add up to a different t-vector. Fix any random non-overlapping choice for the first two t-vectors, and consider the probability that the third resides fully in these 2t columns (but does not equal any of the first two vectors). The latter probability is $\Theta((2t/N)^t)$. Considering all $\binom{q}{3}$ choices of the rows, the claim follows.

UNIVERSAL HASH BASED MACS. We now discuss the application to message authentication. Let D be some domain consisting of messages we want to authenticate. (For example D could be $\{0,1\}^*$, or all strings of length up to some maximum length.) We fix a family H of ϵ -AXU hash functions in which each function $h \in H$ maps from D to $\{0,1\}^n$. We also let F be a family of functions with domain $\{0,1\}^n$ and range $\{0,1\}^m$.

The standard paradigm is that to authenticate message $M \in D$, pick a value $r \in \{0,1\}^n$ and set the mac to $(r,f(r)\oplus h(M))$. Here $\langle h,f\rangle$ is the (secret) key under which macs are created and verified, where $h \in H$ and $f \in F$. The counter version sets r to a counter value that is incremented with each message authenticated. Denoting it by StandardMAC-Ctr,

$$\begin{split} &\mathbf{InSec}^{\mathrm{mac}}(\mathsf{StandardMAC-Ctr}, q_a, q_v, T) \\ &\leq q_v \epsilon + \mathbf{InSec}^{\mathrm{prf}}(F, q_a + q_v, T') \;. \end{split}$$

where $q_a < N$, $q_v \ge 1$, $N = 2^n$ and $T' = T + O((q_a + q_v)(n + m))$. When a stateless scheme is desired, the standard paradigm would pick r at random. A chosen-message attack of q messages results in a collision in r values with probability $\Theta(q^2/N)$, and when this happens forgery is possible. We wish to apply the parity construct to get better security, comparable or superior to that of the counter version.

OUR SCHEME. The idea is that instead of picking one point r, the generator of the mac picks t distinct random points r_1, \ldots, r_t , and sets the mac of M to $(r_1, \ldots, r_t, f(r_1) \oplus \cdots \oplus f(r_t) \oplus h(M))$, the setting being the same as above.

More precisely, with H fixed we associate to F a message authentication scheme $\mathsf{MACRX}_t[F]$, parameterized by the integer $t \geq 1$. It consists of two algorithms, one to generate macs, and the other to verify candidate macs. (The distinction is necessary since the mac generation algorithm is probabilistic.) These algorithms are described in Figure 4. The mac generation algorithm takes as input a key $\langle h, f \rangle$ and a message $M \in D$, while the verification algorithm takes the same key, a message, and a candidate mac for it. Here h is a random hash

Fig. 4. MACRX_t[F]: Our message authentication scheme: Here $M \in D$ is the text to be authenticated and $\langle h, f \rangle \in H \times F$ is the key.

function from H while f is a random member of F. It is understood that f is accessible as an oracle. (When F is pseudorandom, a seed explicitly supplied to the algorithms names a particular function in the family and thus enables computation of the oracle. But the view of f as an oracle better suits the analysis.)

We stress one aspect of the verification procedure, namely to check that the candidate tag really contains t points (not more or less) and that these are distinct. Without this check, forgery is possible.

The security of our scheme can be analyzed via a connection to matrix vulnerability and Lemma 4, as detailed in [4], to yield the following.

Theorem 2. Let H be a family of ϵ -AXU hash functions with domain D and range $\{0,1\}^n$. Let F be a family of (pseudorandom) functions with domain $\{0,1\}^n$ and range $\{0,1\}^m$. Let $N=2^n$ and assume t is an odd integer satisfying $1 \le t \le \sqrt{N/(2e \lg N)}$. Let $\mathsf{MACRX}_t[F]$ be the associated MAC as defined above. Assume $1 \le q_a \le N/(2e^2t)$ and $q_v \ge 1$. Then

$$\begin{split} \mathbf{InSec}^{\mathrm{mac}}(\mathsf{MACRX}_t[F], q_a, q_v, T) \leq \\ q_v \epsilon + d'(t, n) \cdot \frac{q_a^3}{N^t} + \mathbf{InSec}^{\mathrm{prf}}(F, t(q_a + q_v), T') \;, \end{split}$$

where $T' = T + O(t(q_a + q_v)(n + m))$ and d'(t, n) is as in Equation (13).

Thus, $\mathsf{MACRX}_3[F]$ offers better security than $\mathsf{MACRX}_1[F]$, and for $q_a < 2^{2n/3}$ its security is comparable to the counter-version as given in Equation (14). $\mathsf{MACRX}_5[F]$ is comparable in security to the counter-version.

Acknowledgments

We thank R. Venkatesan and B. Yee for helpful discussions on the subject of the uses and drawbacks of counters in computer systems.

Mihir Bellare was supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering. Hugo Krawczyk was supported by the Fund for the Promotion of Research at the Technion.

References

- 1. W. AIELLO, AND R. VENKATESAN. Foiling birthday attacks in length-doubling transformations. Advances in Cryptology Eurocrypt 96 Proceedings, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.
- 2. M. Bellare, A. Desai, E. Jokipii and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE, 1997.
- 3. M. Bellare, O. Goldreich and H. Krawczyk. Beyond the birthday barrier, without counters. Full version of this paper, available via http://www-cse.ucsd.edu/users/mihir.
- 4. M. BELLARE, R. GUÉRIN AND P. ROGAWAY. XOR MACs: New Methods for Message Authentication using Finite Pseudorandom Functions. Full version available via http://www-cse.ucsd.edu/users/mihir. Preliminary version in Advances in Cryptology Crypto 95 Proceedings, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
- M. Bellare, J. Kilian and P. Rogaway. The Security of Cipher Block Chaining. Advances in Cryptology Crypto 94 Proceedings, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.
- M. Bellare, T. Krovetz and P. Rogaway. Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible. Advances in Cryptology Eurocrypt 97 Proceedings, Lecture Notes in Computer Science Vol. 1233, W. Fumy ed., Springer-Verlag, 1997.
- 7. O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217.
- 8. C. Hall, D. Wagner, J. Kelsey and B. Schneier. Building PRFs from PRPs. Advances in Cryptology Crypto 98 Proceedings, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
- H. Krawczyk. LFSR-based Hashing and Authentication. Advances in Cryptology Crypto 94 Proceedings, Lecture Notes in Computer Science Vol. 839,
 Y. Desmedt ed., Springer-Verlag, 1994.
- M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Computing, Vol. 17, No. 2, April 1988.
- 11. M. NAOR AND O. REINGOLD. On the construction of pseudorandom permutations: Luby-Rackoff revisited. J. of Cryptology Vol. 12, No. 1, 1999, pp. 29–66.
- J. PATARIN. Improved security bounds for pseudorandom permutations. Proceedings of the Fourth Annual Conference on Computer and Communications Security, ACM, 1997.
- 13. J. PATARIN. About Feistel schemes with six (or more) rounds. *Proceedings of the 5th Fast Software Encryption Workshop*, Lecture Notes in Computer Science Vol. 1372, Springer-Verlag, 1998.
- 14. B. Preneel and P. van Oorschott. MDx-MAC and building fast MACs from hash functions. Advances in Cryptology Crypto 95 Proceedings, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
- V. Shoup. On Fast and Provably Secure Message Authentication Based on Universal Hashing. Advances in Cryptology Crypto 96 Proceedings, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
- M. WEGMAN AND L. CARTER. New hash functions and their use in authentication and set equality. J. of Computer and System Sciences, vol. 22, 1981, pp. 265-279.

Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto'97

Phong Nguyen

École Normale Supérieure, Laboratoire d'Informatique 45 rue d'Ulm, 75230 Paris Cedex 05, France Phong.Nguyen@ens.fr http://www.dmi.ens.fr/~pnguyen/

Abstract. Recent results of Ajtai on the hardness of lattice problems have inspired several cryptographic protocols. At Crypto '97, Goldreich, Goldwasser and Halevi proposed a public-key cryptosystem based on the closest vector problem in a lattice, which is known to be NP-hard. We show that there is a major flaw in the design of the scheme which has two implications: any ciphertext leaks information on the plaintext, and the problem of decrypting ciphertexts can be reduced to a special closest vector problem which is much easier than the general problem. As an application, we solved four out of the five numerical challenges proposed on the Internet by the authors of the cryptosystem. At least two of those four challenges were conjectured to be intractable. We discuss ways to prevent the flaw, but conclude that, even modified, the scheme cannot provide sufficient security without being impractical.

1 Introduction

Historically, public-key cryptosystems have almost without exception been built on the assumed hardness of one of the three following problems: knapsacks, discrete logarithms in some groups, and integer factorization. Despite the NP-completeness of the knapsack problem, all knapsack-based systems have been broken (see the survey [24]), mainly due to the connection between lattice problems and knapsacks arising from cryptography. The narrowness of the remaining options has often been cited as a potential fragility of public-key cryptography. Recently, Ajtai [1] found a surprising worst-case/average-case connection for certain lattice problems, which caused a revival of knapsack-based cryptography [3,16,15,19,5]. In particular, two lattice-based public-key cryptosystems have received wide attention: the Ajtai-Dwork cryptosystem (AD) [3] and the Goldreich-Goldwasser-Halevi cryptosystem (GGH) [16].

The AD scheme has a fascinating property: it is provably secure unless some worst-case lattice problem can be solved in probabilistic polynomial time. The problem is a variant of the famous shortest vector problem (SVP), which refers to the question of computing a lattice vector with minimum non-zero Euclidean

length. The GGH scheme relies on the non-homogeneous analog of SVP, the socalled closest vector problem (CVP) in which one has to find a lattice vector minimizing the distance to a given vector. GGH has no proven worst-case/averagecase property, but it is much more practical than AD. Specifically, for security parameter n, key-size and encryption time are $O(n^2)$ for GGH, vs. $O(n^4)$ for AD. For RSA and El-Gamal systems, key size is O(n) and computation time is $O(n^3)$. The authors of GGH argued that the increase in size of the keys was more than compensated by the decrease in computation time.

Ajtai's work [1] initiated a substantial amount of research [7,2,22,13,9,6,17] on the hardness of SVP, CVP, and related problems. We now know that SVP is NP-hard for polynomial random reductions [2], even up to some constant [22]. CVP is NP-hard [11], and approximating CVP to within almost-polynomial factors is also NP-hard [9]. In fact, SVP cannot be harder than CVP: it was recently shown [17] that one can approximate SVP to any factor in polynomial time given an approximation CVP-oracle for the same factor and dimension. On the other hand, approximating SVP or CVP to $\sqrt{n/\log(n)}$ are unlikely to be NP-hard [13] (n denotes the lattice dimension). Furthermore, there exist polynomial-time approximation algorithms [20,4,25,27], such as the celebrated LLL algorithm, that can achieve exponential bounds. And it is well-known that these reduction algorithms behave much better in practice than theoretically expected. Therefore, the practical security of AD and GGH had to be assessed.

In the case of AD, such an assessment proved to be deadly: Nguyen and Stern [23] showed that any realistic implementation of the AD scheme was insecure. However, the attack was specific to AD, and had no implications on the security of GGH. Moreover, since GGH is much more efficient than AD, it has larger security parameters: breaking GGH apparently meant solving hard lattice problems in dimensions much higher than what had previously been done, such as in cryptanalyses of knapsack systems. Based on numerous experiments, the authors of GGH conjectured that the closest vector problem arising from their scheme was intractable in practice for dimension 300 or so. To bring confidence in their scheme, they published on the Internet a series of five numerical challenges [14], in dimensions 200, 250, 300, 350 and 400. In each of these challenges, a public key and a ciphertext were given, and the challenge was to recover the plaintext.

To our knowledge, all previous attempts to solve these challenges have failed, except in dimension 200. The most successful attempts used the so-called embedding technique, which (heuristically) reduces CVP to a shortest vector problem in a lattice of similar dimension. By applying high-quality reduction algorithms, one hopes to recover the closest vector. Using this technique, Schnorr *et al.* [26] were able to decrypt ciphertexts up to dimension 150. And Nguyen used the NTL package in Oct. 97 to solve the 200-challenge in a few days (see the documentation of NTL [28]) with the improved algorithm of [27]. But higher dimensions seemed to be out of reach, confirming the predictions of the authors of GGH.

In this paper, we show that there is dangerous flaw in the GGH cryptosystem. More precisely, we observe that each encryption leaks information on the

cleartext, and this information leakage allows an attacker to reduce the problem of decrypting ciphertexts to solving particular CVP-instances which are much easier than the general problem. Namely, for these instances, the given vector is very close to the lattice, which makes it possible in practice to find the closest vector by standard techniques. As an application, we solved four out of the five Internet GGH challenges from dimension 200 to 350, in a reasonable time. In dimension 400, we obtained 1/8th of the plaintext. This proves that GGH is insecure for the parameters suggested by Goldreich, Goldwasser and Halevi. Learning the result of our experiments, one of the authors of GGH declared the scheme as "dead" [12]. We suggest modifications to fix the encryption process, but estimate that, even modified, the scheme cannot provide security without being impractical, compared to existing schemes.

The rest of the paper is organized as follows. We first review necessary material about lattices in section 2. Then we briefly describe the GGH cryptosystem. In section 4, we explain how the encryption process leaks information, using a basic observation. In section 5, we show how to exploit the information leakage to simplify the problem of decrypting ciphertexts. Section 6 presents the experiments done on the Internet challenges. And in section 7, we discuss ways to repair the scheme.

2 Background on Lattices

In the sequel, we denote vectors by bold-face lowercase letters $(e.g. \mathbf{b}, \mathbf{c}, \mathbf{r})$; we use capital letters (e.g. B, C, L) to denote matrices or sets of vectors. If \mathbf{b} is a vector in \mathbb{R}^n , then $\lceil \mathbf{b} \rfloor$ denotes the vector in \mathbb{Z}^n which is obtained by rounding each entry in \mathbf{b} to the nearest integer.

2.1 Definitions

In this paper, we only care about integral lattices of full rank, so the definitions below only deal with those. Let M be a non-singular $n \times n$ integral matrix. Denote by $\mathbf{b}_1, \dots, \mathbf{b}_n$ the row vectors of M. The lattice L spanned by $(\mathbf{b}_1, \dots, \mathbf{b}_n)$ (or M) is the set L(M) of all integral linear combinations of the \mathbf{b}_i 's. The set of \mathbf{b}_i 's is called a basis of L. We identify a basis with the square matrix whose rows are the basis vectors (note that the convention of [16] used columns instead of rows). Naturally, a lattice is a set $L \subset \mathbb{Z}^n$ for which there exists a non-singular matrix M such that L = L(M). For a given lattice L, there exist many bases, which all differ by multiplication with some unimodular matrix. Thus, all bases have the same determinant in absolute value, which is called the lattice determinant det(L). In every lattice L, there is a non-zero vector whose Euclidean length is $O(\sqrt{n}\det(L)^{1/n})$. In a "random" lattice, one generally assumes that there are no non-zero vectors with substantially shorter length. The goal of lattice reduction is to find a reduced basis, that is, a basis consisting of reasonably short vectors. Define the lattice gap as the ratio between the second successive minimum (the smallest real number r such that there are two linearly independent lattice points of length at most r) and the length of a shortest non-zero vector. Experiments suggest that the larger the lattice gap is, the easier reduction becomes.

2.2 Algorithmic Problems

We recall that the shortest vector problem (SVP) is: given a lattice basis, find a non-zero lattice vector with minimal Euclidean length. The closest vector problem (CVP) is: given a lattice basis and a vector $\mathbf{c} \in \mathbb{Z}^n$, find a lattice vector which minimizes the distance to \mathbf{c} . A related problem is the smallest basis problem (SBP): given a lattice basis, find a basis which minimizes the product of the lengths of its elements. All these problems are NP-hard. And no polynomial-time algorithm is known for approximating either SVP, CVP or SBP in \mathbb{Z}^n to within a polynomial factor in n. In fact, the existence of such algorithms is an important open problem. The best polynomial time algorithms achieve exponential factors, and are based on the LLL algorithm [20]. The LLL algorithm can be viewed as an approximation algorithm for both SVP and SBP. Variants [25,27] of LLL achieve better approximation factors for SVP, but with slower running time. More precisely, Schnorr defined a family of algorithms (BKZ [25]) whose performances depend on a parameter called the blocksize. These algorithms use some kind of exhaustive search which is exponential in the blocksize. So far, the best reduction algorithms in practice are variants [27] of those BKZ-algorithms, which apply a heuristic to reduce exhaustive search.

Babai [4] showed how to use a reduced basis to approximate CVP. The more reduced the basis is, the better the approximation is. For an LLL-reduced basis, this yields an exponential factor. But in practice, the best method to solve CVP is the so-called embedding technique (see [16]), which reduces the problem to a shortest vector problem. Let $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a basis of a lattice L, and \mathbf{c} the given vector corresponding to a CVP-instance. The embedding technique builds the lattice in \mathbb{Z}^{n+1} spanned by the rows of the following matrix:

$$L' = \begin{pmatrix} -\mathbf{b}_1 - 0 \\ -\vdots - 0 \\ -\mathbf{b}_n - 0 \\ -\mathbf{c} - 1 \end{pmatrix}$$

L' and L have the same determinant and almost the same dimension, therefore one would expect that the shortest lattice vector of L' has about the same length than the shortest lattice vector of L. Now, assume that the vector $\mathbf{v} \in L$ minimizes the distance to \mathbf{c} . One can see that the vector $(\mathbf{c} - \mathbf{v}, 1) \in \mathbb{Z}^{n+1}$ is short and belongs to L'. In fact, one hopes that it is the shortest vector of L', so that one solves the CVP-instance from the SVP-instance defined by L'. Clearly, this embedding method is heuristic. Still, if \mathbf{c} is very close to the lattice L, then $\mathbf{c} - \mathbf{v}$ will be much shorter than the shortest vector of L, so that L' has a large gap, and hopefully, is easy to reduce.

3 The Goldreich-Goldwasser-Halevi Cryptosystem

We make a brief description of the GGH Cryptosystem. More details can be found in [16]. In particular, we do not specify the key generation because it is unnecessary for the understanding of our attack. Roughly speaking, GGH is the lattice-analog of the McEliece [21] cryptosystem based on algebraic coding theory. In both schemes, a ciphertext is the addition of a random noise vector to a vector corresponding to the plaintext. The public key and the private key are two representations of the same object (a lattice for GGH, a linear code for McEliece). The private key has a particular structure which allows to suppress noise vectors up to a certain bound. However, the domains in which all these operations take place are vastly different.

The security parameter is $(n, \sigma) \in \mathbb{N}^2$. The integer n is much larger than σ : a typical value is $(n, \sigma) = (300, 3)$. A lattice L in \mathbb{Z}^n is generated together with a reduced basis R of L. One actually generates a non-singular matrix R with short row vectors, and defines the lattice spanned by this matrix. The basis R, which is kept private, is transformed to a non-reduced basis R, which will be public. Several transformation methods were proposed in [16]. Roughly speaking, these methods multiply R by sufficiently many small unimodular matrices. Computing a basis as "good" as the private basis, given only the non-reduced basis, means approximating SBP.

The message space is a "large enough" parallelepiped in \mathbb{Z}^n . For instance, in the numerical challenges, cleartexts were randomly chosen in $[-128\cdots+127]^n$ so that they could be stored within 8n bits. A message $\mathbf{m} \in \mathbb{Z}^n$ is encrypted into $\mathbf{c} = \mathbf{m}B + \mathbf{e}$ where \mathbf{e} is an error vector uniformly chosen from $\{-\sigma, \sigma\}^n$. In [16], other methods to embed messages into lattice points are discussed. Note, that as it stands, the cryptosystem is not semantically secure, because one can check if a ciphertext \mathbf{c} corresponds to a plaintext \mathbf{m} by computing $\mathbf{c} - \mathbf{m}B$. A ciphertext \mathbf{c} is decrypted as $[\mathbf{c}R^{-1}]RB^{-1}$ (note: this is Babai's round method [4] to solve CVP). The private basis R is generated in such a way that the decryption process succeeds with high probability. More precisely, the following result is proved in [16]:

Theorem 1. Let R be the private basis, and denote the maximum L_{∞} norm of the columns in R^{-1} by γ/\sqrt{n} . Then the probability of decryption errors is bounded by $2ne^{-1/(8\sigma^2\gamma^2)}$.

The larger σ is, the harder the CVP-instances are expected to be. But the previous theorem shows that σ must be small for the decryption process to succeed. In practice, one chooses a private basis R, looks at the norm of the columns in R^{-1} , and then takes the maximal possible value for σ . The authors of GGH considered that the value $\sigma = 3$ was a good compromise. Based on extensive experiments (see [16]) with all known methods to solve CVP, they conjectured that the cryptosystem was secure in practice for dimension 300 or so. The most successful attack (embedding technique using improved reduction algorithm) against GGH could decrypt ciphertexts up to dimension 200, in about a few days.

We now present new results on the security of GGH, which show that GGH cannot provide sufficient security even for dimensions as high as 400. In fact, there is an intrinsic weakness in the encryption process which will prove very dangerous.

4 Leaking Remainders

Let (n, σ) be the security parameter, and B be a public basis. Assume that a message $\mathbf{m} \in \mathbb{Z}^n$ is encrypted into a ciphertext $\mathbf{c} \in \mathbb{Z}^n$ with B. There is a vector $\mathbf{e} \in \{\pm \sigma\}^n$ such that:

$$\mathbf{c} = \mathbf{m}B + \mathbf{e} \tag{1}$$

We will see that this equation defining the encryption process has a major flaw. The key to our results is to look at (1) modulo some well-chosen integer. By an appropriate choice of the modulus, the error vector \mathbf{e} will disappear, from which information on \mathbf{m} can be derived. Since each entry of \mathbf{e} is $\pm \sigma$, a natural candidate is σ , which gives: $\mathbf{c} \equiv \mathbf{m}B \pmod{\sigma}$. But we notice that 2σ is a better modulus. Indeed, if we let $\mathbf{s} = (\sigma, \dots, \sigma) \in \mathbb{Z}^n$, then we have $\mathbf{e} + \mathbf{s} \equiv 0 \pmod{2\sigma}$, so that:

$$\mathbf{c} + \mathbf{s} \equiv \mathbf{m}B \pmod{2\sigma} \tag{2}$$

This reads as a modular system in the unknown \mathbf{m} . If we can solve this system, we obtain \mathbf{m} modulo 2σ , which we denote by $\mathbf{m}_{2\sigma}$. Two questions arise: how many solutions are there? And how can we compute all of them? We will see that with high probability, there are very few solutions, which are easy to compute. With non-negligible probability, there is even a single solution. The probability is with respect to B. Since B is obtained by randomly mixing a private basis R, and since 2σ is a small number, we assume that the entries of B modulo 2σ are uniformly and independently distributed in $\mathbb{Z}_{2\sigma}$. Recall that, in practice, the standard choice is $\sigma = 3$, so that $2\sigma = 6$.

We now discuss the general problem of solving a linear system $\mathbf{y} = \mathbf{x}B \pmod{N}$, where the vector \mathbf{y} , the (random) matrix B and the (small) modulus N are known and we know that there is at least one solution. Clearly, two solutions differ by an element of the kernel of B, defined as the set of $\mathbf{x} \in \mathbb{Z}^n$ such that $\mathbf{x}B \equiv 0 \pmod{N}$. It follows that all solutions can be found from the kernel of B and a particular solution to the system, and their number is equal to the cardinal of the kernel. If the matrix B is invertible modulo N, that is, if $\det(B)$ is coprime to N, then there is only one solution, which can be found by matrix inversion: $\mathbf{x} = \mathbf{y}B^{-1} \pmod{N}$. This is of course the simplest case. Let us see how often such a case occurs.

4.1 Invertible Matrices

The material covered in sections 4.1 and 4.2 is quite intuitive and is probably already known. Since we have not been able to locate appropriate references,

we provide proofs in the appendix. The following result gives the proportion of invertible matrices modulo N among all matrices:

Theorem 2. Let N be some positive integer. Let p_1, \ldots, p_ℓ be the distinct prime factors of N. Consider the ring of $n \times n$ matrices with entries in \mathbb{Z}_N . Then the proportion of invertible matrices (i.e., with determinant coprime to N) is equal to:

$$\prod_{i=1}^{\ell} \prod_{k=1}^{n} (1 - p_i^{-k}).$$

Note that the above proportion converges rapidly to its limit. It follows that the proportion can be considered as constant for dimensions of interest, since those dimensions are high (higher than 200). Table 1 gives numerical results. It shows that with non-negligible probability, the public matrix B is invertible

Table 1. The proportion of invertible matrices modulo N.

Modulus N	2	3	4	5	6	7	8	9	10
%	28.9	56.0	28.9	76.0	16.2	83.7	28.9	56.0	22.0

modulo 2σ , which discloses any plaintext modulo 2σ . We now show that when the matrix is not invertible, its kernel is most of the time very small.

4.2 Matrices with Small Kernel

We first treat the case of a prime modulus p. The kernel is then a \mathbb{Z}_p -vector space. If d is the kernel dimension, the number of solutions is p^d . If both p and d are small, this number is small. The following theorem shows that the vast majority of non-invertible matrices modulo p have a kernel of dimension less or equal to two:

Theorem 3. Let \mathbb{F}_q be the finite field with q elements, where q is a prime power. Consider the set of $n \times n$ matrices with entries in \mathbb{F}_q . We have:

1. The proportion of matrices with one-dimensional kernel is equal to:

$$\frac{q}{(q-1)^2}(1-q^{-n})\prod_{k=1}^n(1-q^{-k}).$$

2. The proportion of matrices with two-dimensional kernel is equal to:

$$\frac{q^2}{(q-1)^2(q^2-1)^2}(1-q^{1-n}-q^{-n}+q^{1-2n})\prod_{k=1}^n(1-q^{-k}).$$

Table 2. The proportion of square modular matrices of low-dimensional kernel.

Again, we note that the above proportions converge quickly to their limit, so that the numerical results of table 2 hold for any sufficiently high dimension. It shows that with high probability, the kernel has dimension less than 2, which means that there are most p^2 solutions to the modular system. Furthermore, it is very simple to compute these solutions. One can compute a kernel basis of any matrix modulo any prime in polynomial time (see [8]). To find a particular solution of the system, one can build a $(n+1) \times (n+1)$ matrix from the image vector \mathbf{y} and the matrix B, as in the embedding technique. It can be shown that a particular solution can be derived from any kernel basis of the new matrix. We refer to [8] for more details.

Now, if the modulus N is not prime but square-free (for instance, 6), the previous results allow us to conclude. Indeed, if $N = p_1 \times \cdots \times p_\ell$ where the p_i 's are distinct primes, then all the solutions can be recovered by Chinese remainders from the solutions modulo each prime p_i . And the total number of solutions is obtained by multiplying the number of solutions for each prime. Furthermore, one can compute the proportion of matrices with respect to the number of elements of their kernel, from the previous proportions. Table 3 gives the results for a modulus equal to 6, which is the case of interest. It shows that only a very small minority of matrices modulo 6 have a kernel with more than 12 elements. The most probable cases are 2, 6, 1 and 3 elements. Otherwise,

Table 3. The proportion of square matrices modulo 6 with small kernel

Kernel cardinal	1	2	3	4	6	9	12	18	36	Other cases
%	16.2	32.4	12.1	7.2	24.3	0.6	5.4	1.1	0.3	0.6

if the modulus N is not square-free, the previous methods do not apply. Still, there exist methods to solve such modular systems, as a method is implemented in the Lidia package [18]. This implementation can compute all the solutions to a linear system modulo any small composite number, in about the same time required for a prime modulus. We have not been able yet to locate further references, but we believe such a classical problem has already been studied. We also believe that the number of solutions is still small most of the time, although we do not know whether this number can easily be estimated. The problem is that we have \mathbb{Z}_N -modules instead of vector spaces. The ring \mathbb{Z}_N has zero divisors,

and all its ideals are principal ideals. The modules will not usually be free modules, so counting the number of elements is harder. We stress that the case of non-square-free numbers does not occur for the suggested choice of parameters in GGH. And if one selects different parameters for which the modulus has a square factor, then one can apply the mentioned algorithm and still hope that the number of solutions is small.

4.3 Security Implications

We saw that for the suggested choice of parameters, the public matrix B had very small kernel with high probability. Then, for any ciphertext \mathbf{c} , the underlying linear system has very few solutions. It follows that, even though the encryption method is probabilistic, one can check whether a given ciphertext corresponds to a given plaintext (knowing only a small fraction of the plaintext), or whether two given ciphertexts correspond to the same plaintext, with overwhelming probability. Thus, the GGH cryptosystem is far from being semantically secure. One might argue that other methods of embedding plaintexts into lattice points were proposed in [16] to make sure the scheme was semantically secure. But the only other practical method suggested to embed the message in the least-significant bits of the coordinates. Namely, instead of picking $\mathbf{m}B$ as the lattice point to be perturbed, it was suggested to pick $\mathbf{v}B$ where the message would form the least-significant bits of \mathbf{v} 's entries, and the remaining bits would be chosen at random. The method is even worse, since our techniques recover the remainders of \mathbf{v} , that is, the least-significant bits. We will discuss ways to fix the flaw in the encryption process in section 7.

5 Simplifying the Closest Vector Problem

Assume now that one knows the plaintext \mathbf{m} modulo 2σ which we denote by $\mathbf{m}_{2\sigma}$. We explain how this partial information simplifies the decryption problem. Recall that we have:

$$\mathbf{c} = \mathbf{m}B + \mathbf{e}$$
.

Therefore:

$$\mathbf{c} - \mathbf{m}_{2\sigma} B = (\mathbf{m} - \mathbf{m}_{2\sigma}) B + \mathbf{e}.$$

The vector $\mathbf{m} - \mathbf{m}_{2\sigma}$ is of the form $2\sigma \mathbf{m}'$ where $\mathbf{m}' \in \mathbb{Z}^n$. It follows that:

$$\frac{\mathbf{c} - \mathbf{m}_{2\sigma}B}{2\sigma} = \mathbf{m}'B + \frac{\mathbf{e}}{2\sigma}.$$

The rational point $\frac{\mathbf{c}-\mathbf{m}_{2\sigma}B}{2\sigma}$ is known, so that the previous equation reads as a closest vector problem for which the error vector $\mathbf{e}/(2\sigma) \in \{\pm \frac{1}{2}\}^n$ is much smaller. The error vector length is now $\sqrt{n/4}$, compared to $\sigma\sqrt{n}$ previously. And if one can solve the new CVP-instance, one can easily solve the former CVP-instance, due to the relationship between the two error vectors. In other

words, we have reduced the problem of decrypting ciphertexts (that is, CVP-instances for which the error vector has entries $\pm \sigma$), to a simpler CVP-problem for which the error vector has entries $\pm \frac{1}{2}$. At this point, we note that there might exist specialized CVP-algorithms to solve such CVP-instances. But we can also apply traditional methods such as the embedding technique, which are more likely to work now that the error vector is smaller.

The previous section showed that one was not guaranteed to obtain $\mathbf{m}_{2\sigma}$ (the message modulo 2σ). However, with high probability, one can confine $\mathbf{m}_{2\sigma}$ to a very small set which can be computed. It follows that the general problem of decrypting ciphertexts can be reduced to solving a very small number of CVP-instances, among which one is easier than the former CVP-instance. Note that it is not necessary to solve all the CVP-instances: it suffices to solve the right one. In practice, this means reducing several lattices in parallel until one of them provides the solution.

The fact that the new CVP-instance involves rational points is not a problem. Rationals can be avoided by multiplying by two the equation defining the CVP-instance. One thus obtains an integer CVP-instance for which the error vector has entries ± 1 and the lattice is the original lattice with doubled entries.

Another way to avoid rational arithmetic is to make the error vector a multiple of 2σ by translation. For instance, letting again $\mathbf{s} = (\sigma, \dots, \sigma) \in \mathbb{Z}^n$, we have: $\mathbf{c} + \mathbf{s} - \mathbf{m}_{2\sigma}B = (\mathbf{m} - \mathbf{m}_{2\sigma})B + \mathbf{e} + \mathbf{s}$. The vector $\mathbf{e} + \mathbf{s}$ is of the form $2\sigma\mathbf{e}'$, where $\mathbf{e}' \in \{0,1\}^n$. It follows that the vector $\mathbf{c} + \mathbf{s} - \mathbf{m}_{2\sigma}B$ is of the form $2\sigma\mathbf{c}'$, where $\mathbf{c}' \in \mathbb{Z}^n$. Therefore, dividing by 2σ , one gets: $\mathbf{c}' = \mathbf{m}'B + \mathbf{e}'$. This is a closest vector problem for which the error vector has expected length $\sqrt{n/2}$, which is slightly worse than $\sqrt{n/4}$.

6 Experiments on the Internet Challenges

To validate our results, we tested the method on the Internet challenges [14] provided by the authors of the cryptosystem, because the hardness of the CVP-instances partly depends on the way the private basis is transformed to the public basis. For each challenge, the cleartext was a vector of \mathbb{Z}^n with entries in $[-128\cdots+127]$, and σ was equal to 3. The private basis was generated by multiplying the $n\times n$ identity matrix by $4\lceil \sqrt{n}+1 \rceil$, and then adding to each entry a random integer chosen from $[-4\cdots+3]$. Although we do not know the private basis, we can estimate the length of the shortest vector by $\sqrt{(4\lceil \sqrt{n}+1 \rceil)^2+(n-1)\times 11/2}$. This allows to estimate the gap of the embedded lattice, so that we get a feeling on the hardness of the SVP-instance defined by the embedding attack. The larger the gap is, the easier the reduction is expected to be.

We implemented the attack using the NTL library [28] developed by Victor Shoup. Timings are given for a 500-MHz 64-bit DEC Alpha running on Linux. For all experiments, we had to use the so-called Givens floating-point variants of reduction algorithms, which are slower than the standard floating-point variants but less prone to stability problems. The results are summarized in table 4 and

5. Table 4 refers to an error vector in $\{\pm \frac{1}{2}\}^n$, whereas table 5 refers to an error vector in $\{0,1\}^n$. In each dimension, we first solved the linear system modulo 6 described in section 4. The number of solutions is indicated in the corresponding row. It is worth noting that these numbers correspond to what was theoretically predicted by table 3 of section 4. For each solution, we built the corresponding CVP-instance described in section 5. We tried to solve these CVP-instances using the embedding technique and improved reduction algorithms. The Expected gap row gives an approximation of the embedded lattice gap. Of course, all the reductions were performed in parallel: each workstation took care of a single CVP-instance, and the running time is with respect to the workstation who found the solution. In all our experiments, at most six workstations were required at the same time. The reduction process was the following: apply a LLL reduction; then a BKZ reduction [25] with blocksize 20; and if necessary, a pruned-BKZ reduction [27] with blocksize 60 and pruning factor 14. From dimension 200 to 300, the embedding used the public basis: it is likely that the running times could have been decreased if we had started after reduction of the public basis. Therefore, the running times should not be considered as optimal. We stress that it is not necessary to perform a complete BKZ-reduction: one just keeps reducing until the correct solution has been found (that is the time indicated in the Time row). In our experiments, the ratio between the total reduction time and the actual time for which the basis was sufficiently reduced to provide the solution is around 3. Thus, stopping the reduction as soon as the solution is found significantly reduces the running time, which is not very surprising. In dimension 350, the embedding used the public basis after BKZ-reduction with blocksize 20, and the corresponding running time does not include the reduction of the public basis. This is because we were not able to recover the plaintext after 4 days of computation, when the embedding started with the public basis (and not the public basis reduced). We are unable to explain this phenomenon. In fact, the behavior of lattice reduction algorithms is still not very well understood.

Table 4. Experiments on the Internet Challenges, with a $\{\pm \frac{1}{2}\}$ -error.

Dimension	200	250	300	350	400
Number of solutions mod 6	2	2	6	6	1
Expected gap	9.7	9.4	9.5	9.4	9.6
Block size	20	20	20	60	60
Type of reduction	BKZ	BKZ	BKZ	Pruning	Pruning
Time in minutes	30	60	165	270	Unsolved

The results demonstrate the power of the new attack. We are able to solve all challenges except in dimension 400, in a reasonable time, although challenges in dimension 300 and 350 were assumed to be intractable. We even obtained information on the plaintext in dimension 400, since we recovered all the remainders

Dimension	200	250	300	350	400
Expected gap	6.9	6.6	6.7	6.6	6.8
Block size	20	20	20	60	60
Type of reduction	BKZ	BKZ	BKZ	Pruning	Pruning
Time in minutes	30	60	240	1290	Unsolved

Table 5. Experiments on the Internet Challenges, with a $\{0,1\}$ -error.

modulo 6. The 200-challenge took only 30 minutes, whereas previous methods required a few days. Challenges in dimension 250 and 300 take respectively 1 and 3 hours, and these timings are not even optimal. The running time in dimension 350 is much larger than in dimension 300 (if we take into account the reduction of the public basis) because a stronger reduction algorithm was required. It is hard to guess what the required time would be for higher dimensions.

Our results suggest that in order to be secure, GGH would require working in dimensions at least higher than 400. But already in dimension 400, the scheme is not really practical: for the 400-challenge, the public key takes 1.8 Mbytes, and the ciphertext takes 6.4 Kbytes, which represents a message expansion of 16.6. And each encryption requires $400^2 \approx 2^{17}$ multiplications with numbers of bit-length up to 129.

7 Repairing the Scheme

Our attack used two "qualitatively different" weaknesses of the scheme. The first one is inherent to the GGH construction: the error vectors are always quite shorter than the vectors in the lattice. This results in a gap in the embedded lattice, which was exploited in previous embedding attacks (success up to dimension 200). There seems to be no easy way to fix this problem, making CVP-instances arising from GGH easier than general CVP-instances. The second weakness is the particular form of the error vectors in the encryption process. This can be fixed by choosing the error vector \mathbf{e} in such a manner that an attacker no longer knows the value of e modulo some well-chosen integer, and theorem 1 is valid. A careful analysis of the proof of theorem 1 suggests that the theorem remains correct when the entries of the error vector are less than σ in absolute value, and have zero mean. Therefore, the most natural way to prevent the flaw is to choose the entries of the error vector **e** at random in $[-\sigma \cdots + \sigma]$ instead of $\{\pm \sigma\}$. The drawback is that the error vector is smaller now. Indeed, if the entries of e are uniformly chosen from $[-\sigma \cdots + \sigma]$, then the expected length of $\|\mathbf{e}\|$ is approximately $\sigma\sqrt{n/3}$ instead of $\sigma\sqrt{n}$. One can obtain a larger error by choosing the entries at random in $\{\pm \sigma, \pm (\sigma - 1)\}$ instead, but the special form of the vector might be dangerous.

In any case, the error vector is smaller than in the original scheme, which makes the scheme more vulnerable to the first weakness. If we choose the entries in $[-\sigma \cdots + \sigma]$, the gap of the embedded lattice is around 3. Our experiments

showed that similar CVP-instances could be solved by a single computer when the gap was slightly less than 7, up to dimension 350. So it might be reasonable to assume that using larger resources, CVP-instances with a gap of 3 can be solved up to about the same dimension. And if one believes that the problem is much harder, one has to keep in mind that such problems can be solved in dimension 200 with small resources. Unfortunately, GGH would need such small dimensions to hope to be competitive with existing public-key cryptosystems. In fact, in early drafts of GGH [16], the proposed dimension was 150-200. Furthermore, one should not forget about possible improvements over current lattice reduction algorithms. Hence, we feel that it would be dangerous to use a dimension less than 400 even if we fix the flaw, which clearly limits interests in the scheme, even compared to the McEliece cryptosystem. Actually, we do not suggest any precise dimension, because we feel that the lattice reduction area is not understood enough yet.

Finally, we note that the hardness of the GGH CVP-instances is somehow related to the hardness of the problem arising in Ajtai's worst-case/average-case connection [1], and in the AD security proof [3]. The problem is a shortest vector problem in a lattice for which the gap is polynomial in the dimension. Our experiments showed that SVP could effectively be solved for a class of lattices with small gap (around 7), up to dimension 350. Although our class of lattices is particular, it seems to suggest that Ajtai's problem might be tractable up to moderate dimensions. It would be nice to assess the hardness of such problems, both from a theoretical and a practical point of view.

8 Conclusion

We showed that the GGH cryptosystem had a major flaw. The special form of the error vector in GGH CVP-instances is dangerous: partial information on plaintexts can be recovered, and the problem of decrypting ciphertexts can be reduced to solving CVP-instances much easier than the general problem. We demonstrated the effectiveness of these results by solving all the numerical challenges proposed by the authors of the GGH scheme, except in the highest dimension 400. Two of those challenges were conjectured to be intractable. There exist simple ways to prevent the flaw, but even modified, we estimate that the scheme cannot achieve sufficient security without large parameters. Our experiments seem to indicate that, in practice, hard lattice problems can be solved up to high dimensions. We feel that, for the moment, it is risky to speculate on the practical performances of the best lattice-reduction algorithms, because their behavior is still not well understood. Our results suggest that, unless major improvements are found, lattice-based cryptography cannot provide a serious alternative to existing public-key encryption algorithms and digital signatures such as RSA and DSS.

Acknowledgments. We would like to thank the anonymous referees for their helpful comments. We are very grateful to Victor Shoup for developing the Num-

ber Theory Library, and in particular, its efficient and easy-to-use lattice reduction module.

References

- M. Ajtai. Generating hard instances of lattice problems. In Proc. 28th ACM STOC, pages 99–108, 1996. Available at [10] as TR96-007.
- 2. M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proc. 30th ACM STOC*, 1998. Available at [10] as TR97-047.
- 3. M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. 29th ACM STOC*, pages 284–293, 1997. Available at [10] as TR96-065.
- L. Babai. On Lovász lattice reduction and the nearest lattice point problem. Combinatorica, 6:1–13, 1986.
- M. Bellare and D. Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *Proc. of Eurocrypt '97*, volume 1233 of *LNCS*. Springer-Verlag, 1997.
- J. Blömer and J.-P. Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In Proc. 31th ACM STOC, 1999. To appear.
- 7. J.-Y. Cai and A. P. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *Proc. 38th IEEE FOCS*, pages 468–477, 1997.
- 8. H. Cohen. A course in computational algebraic number theory. Springer, 1995. 2nd Edition.
- 9. I. Dinur, G. Kindler, and S. Safra. Approximating-CVP to within almost-polynomial factors is NP-hard. In *Proc. 39th IEEE FOCS*, pages 99–109, 1998. Available at [10] as TR98-048.
- 10. ECCC. http://www.eccc.uni-trier.de/eccc/. The Electronic Colloquium on Computational Complexity.
- P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, Mathematische Instituut, University of Amsterdam, 1981. Report 81-04.
- O. Goldreich. Private communication, Jan. 99.
- O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In Proc. 30th ACM STOC, 1998. Available at [10] as TR97-031.
- 14. O. Goldreich, S. Goldwasser, and S. Halevi. Challenges for the GGH-Cryptosystem. Available at http://theory.lcs.mit.edu/~shaih/challenge.html.
- O. Goldreich, S. Goldwasser, and S. Halevi. Collision-free hashing from lattice problems. Available at [10] as TR96-056., 1996.
- O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Proc. of Crypto'97*, volume 1294 of *LNCS*, pages 112–131. Springer-Verlag, 1997. Available at [10] as TR96-056.
- O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. Available at [10] as TR99-002.
- 18. The LIDIA Group. LiDIA a library for computational number theory. Can be obtained at http://www-jb.cs.uni-sb.de/LiDIA/linkhtml/lidia/lidia.html.
- 19. R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

- A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. Math. Ann., 261:515–534, 1982.
- R.J. McEliece. A public-key cryptosystem based on algebraic number theory. Technical report, Jet Propulsion Laboratory, 1978. DSN Progress Report 42-44.
- D. Micciancio. The shortest vector problem is NP-hard to approximate within some constant. In Proc. 39th IEEE FOCS, 1998. Available at [10] as TR98-016.
- P. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork Cryptosystem. In Proc. of Crypto '98, volume 1462 of LNCS, pages 223–242. Springer-Verlag, 1998.
- A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In Cryptology and Computational Number Theory, volume 42 of Proceedings of Symposia in Applied Mathematics, pages 75–88. A.M.S., 1990.
- C.-P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. Theoretical Computer Science, 53:201–224, 1987.
- C-P. Schnorr, M. Fischlin, H. Koy, and A. May. Lattice attacks on GGH-Cryptosystem. Rump session of Crypto '97.
- C.P. Schnorr and H.H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proc. of Eurocrypt'95*, volume 921 of *LNCS*, pages 1–12. Springer-Verlag, 1995.
- 28. V. Shoup. Number Theory C++ Library (NTL) version 3.6. Can be obtained at http://www.shoup.net/ntl/.

A The Solutions to the Internet Challenges

For completeness, we provide the solutions to the numerical challenges [14] published on the Internet by Goldreich, Goldwasser and Halevi. The message entries are given from left to right, and top to bottom.

A.1 Dimension 200

The message is:

```
-37
                            -65
                                  74
                                      100
                                              38
                                                   71
                                                         87
                                                             110
                                                                  -113
                                                                         -10
                                                                              109
                                                                                     70
                                                                                               116
                                                                                                    -114
                                                                                                                -38
                                                                                                                       32
                                                                                                                                        54
     -46
                      -55
86
     -81
            48
                108
                      -24
                            79
                                  -57
                                       58
                                              25
                                                 -112
                                                      -124
                                                              88
                                                                                     33
                                                                                           64
                                                                                               -19
                                                                                                      49
                                                                                                          -73
                                                                                                                 -38
                                                                                                                       -9
                                                                                                                            92
                                                                                                                                  49
                                                                    90
                                                                        104
                                                                               -1
126
      41
           -90
                 57
                      -80
                            92
                                  16
                                      -33
                                              12
                                                         72
                                                              -36
                                                                           3
                                                                              117
                                                                                     85
                                                                                                                      -47
                                                                                                                                  58 -125
                                                   -3
                                                                    68
                                                                                                      48
                                                                                                                -52
-25 -108
            -5
                -61
                      -28 -127
                                 -62 -115
                                            -89
                                                  124
                                                       -67
                                                             -124
                                                                    13
                                                                         29
                                                                               17
                                                                                   -118
                                                                                          -26
                                                                                               109
                                                                                                      79
                                                                                                          105
                                                                                                                 98
                                                                                                                       36
                                                                                                                            -16
                                                                                                                                  48
                                                                                                                                       -44
                                                       -80
96
    125
            38
               -112
                       55
                            41
                                  -8
                                       76
                                             -91
                                                   60
                                                              -90
                                                                   126
                                                                         -66
                                                                              -50
                                                                                   122
                                                                                           -4
                                                                                               -45
                                                                                                          -102
                                                                                                                100
                                                                                                                       18
                                                                                                                             0
                                                                                                                                  81
                                                                                                                                       -31
      52 -123
                            -38
                                       -30
                                           -128
                                                              42
                                                                    -60
                                                                        102
                                                                               79
                                                                                               118
                                                                                                     127
                                                                                                           -43
                                                                                                                      122
                                                                                         100
      -4 -108
                -20
                      -50
                           -11
                                  86 -125
                                              -6
                                                   48
                                                       -24
                                                              84
                                                                   -21
                                                                         74
                                                                               85
                                                                                    -74
                                                                                                -1
                                                                                                      -5
                                                                                                          74
                                                                                                                 -49
                                                                                                                       -5
                                                                                                                            98
                                                                                                                                 -58
                                                                                                                                        -6
                                                                                                79
          -11
               -118
                      -93
                           118
                                 118
                                       32 -117
                                                  -29
                                                       111
                                                               0
                                                                   -70
                                                                        114 122
                                                                                    106
                                                                                          -38
                                                                                                     -42
                                                                                                          -92
                                                                                                                       12 -120
```

A.2 Dimension 250

The message is:

```
-81 -118
                                                                         -21
 90
     122
            72
                119
                       62
                           108
                                  68
                                       110
                                             119
                                                   73
                                                        -39
                                                              -54
                                                                         -82 -112
                                                                                    -35
                                                                                          -76
                                                                                                95
                                                                                                     112
                                                                                                            31
                                                                                                                 13
                                                                                                                      118
                                                                                                                                 -23
                                                                                                                                       106
                                                                     65
-38
     112
                                -111
                                                         99
                                                               10
                                                                                                75
                                                                                                                                  59
            33
                 -74
                      -86
                             85
                                       -92
                                              29 -120
                                                                     76
                                                                          82
                                                                                1
                                                                                     22
                                                                                          -85
                                                                                                     -40
                                                                                                           -39
                                                                                                                 92
                                                                                                                       53
                                                                                                                             13
                                                                                                                                        38
                                                                                27
                                                                                                                 39
      -56
            29
                       50
                                        34
                                             -88
                                                         77
                                                              125
                                                                                           85
                                                                                                -91
                                                                                                                      100
                                                                               -98 -128
-84
     -50
           -85
                 58
                     -118
                             81
                                 104
                                       -45
                                             -18
                                                  119 -123
                                                              117
                                                                   -32
                                                                          40
                                                                                          -11
                                                                                               -100
                                                                                                      49
                                                                                                           -25
                                                                                                                 55
                                                                                                                      -22
                                                                                                                            -67
                                                                                                                                  93
                                                                                                                                       112
101
          -115
                 56
                      119
                                 101
                                       -58
                                              45
                                                   32
                                                        -48
                                                               -1
                                                                     8
                                                                          35
                                                                               110
                                                                                      0
                                                                                           41
                                                                                                100
                                                                                                      96
                                                                                                           -47
                                                                                                                -126
                                                                                                                       96
                                                                                                                             71
                                                                                                                                  -98
                                                                                                                                        18
-82
      86
          124
                 -20
                       51
                            109
                                  -46
                                       -73
                                             123 -
                                                  117
                                                        -81
                                                                         -11
                                                                               -85
                                                                                           69
                                                                                                43
                                                                                                    -103
                                                                                                                 25
                                                                                                                       26
                                                                                                                          -110
                           110
-28
       0
           -37
                -85
                       31
                                 -38
                                       -11
                                             106
                                                   70
                                                        -88
                                                               87
                                                                  -103
                                                                         -32
                                                                                82
                                                                                     -92
                                                                                           15
                                                                                                -48 -108 -124
                                                                                                                  -4
                                                                                                                       38
                                                                                                                             74
                                                                                                                                 -88
                                                                                                                                        64
-76
      65
           -38
                 -58
                      -65
                             85
                                  42
                                        64
                                             -79
                                                   85
                                                        -33
                                                              -97
                                                                    -81
                                                                          84 -118
                                                                                    118
                                                                                          124
                                                                                                98
                                                                                                    -113
                                                                                                           -35
                                                                                                                 52
                                                                                                                      -77
                                                                                                                            108
                                                                                                                                 -123
                                                                                                                                        -56
                       42
                             46
                                 111
                                       -20
                                                         44
                                                               94
                                                                    97
                                                                         -20
                                                                                                                       93
                                           -119
                                                                             -112
                                                                                     55
                                                                                                -81
                                                                                                     103
```

A.3 Dimension 300

The message is:

```
-25
     114
           -46
                107
                       90
                            -33
                                -18
                                       -7
                                                 -77
                                                             -6
                                                                 -22 -104
                                                                             48
                                                                                  -55
                                                                                             94
                                                                                                             -72
                                                                                                                  -15
                                                                                                                                    -5
                                                                                                  28
                                                                                                                         87
-124
     -44
            15 -108
                       28
                           110
                                  -4
                                       14
                                            -64
                                                 -24
                                                      -23
                                                             32
                                                                  86
                                                                        99
                                                                            -97 -119
                                                                                        14
                                                                                             26
                                                                                                 -12
                                                                                                         38
                                                                                                             -53
                                                                                                                   62
                                                                                                                         77
                                                                                                                             -87
                                                                                                                                   -38
 47
     -31
            76
                       91
                            -57
                                  10
                                             86
                                                  31
                                                        76
                                                             69
                                                                       -37
                                                                           -122
                                                                                             38
                                                                                                  -37
                                                                                                         40
                                                                                                              70
                                                                                                                  -27
                                                                                                                        -73
                                                                                                                                    89
                                       -80
                                                                -100
                                                                                        69
                                                                                                                             -32
       43
                 42
                       84
                           115
                                 -39
                                      -74
                                                           -122
                                                                  -21
                                                                                                  -89
                                                                                                              39
                                                                                                                        -81
                                                                                                                             -50
                                                                                                                                  -122
 88 -108
           -21 -113
                     -12
                           -59
                                109
                                     -97
                                             93
                                                  24
                                                      116
                                                             81
                                                                 114
                                                                       -86
                                                                             16
                                                                                   82
                                                                                       -68
                                                                                             22
                                                                                                  -67
                                                                                                         -2
                                                                                                             -13
                                                                                                                  -56
                                                                                                                        13
                                                                                                                            -107
                                                                                                                                   -17
                     -117
                           -95
                                             12 -112
                                                                                                        83
                                                                                                                      -112
120
      61
          -102
                -89
                                 128
                                       96
                                                       20
                                                             82
                                                                        52
                                                                             48
                                                                                   21
                                                                                        40
                                                                                                                  -39
                                                                                                                                    23
                                                                 125
                                                                                                            -111
                                                                                                                              40
                           -49 -108
                                                                  -67
 14 -100
            95
               -100
                                      111
                                            -22
                                                 -68
                                                      123
                                                             11
                                                                      -101
                                                                           -104
                                                                                  -51
                                                                                             106
                                                                                                                             -100
                                                                                                                                   -24
 95
    -83
            66
                112
                      86
                           -39
                                 -1
                                     114
                                            -71
                                                  27
                                                      -92
                                                              8
                                                                  48 -109
                                                                            114
                                                                                 109 -114
                                                                                              -2
                                                                                                  -86
                                                                                                       -87
                                                                                                             22
                                                                                                                   -9
                                                                                                                        88
                                                                                                                                    66
 -68
    -100
            34
                -55
                     -72 -117
                                -87
                                      -26
                                            77
                                                -103
                                                      -68
                                                            -90
                                                                      -81
                                                                            -32
                                                                                  -76
                                                                                       -45
                                                                                             105
                                                                                                       -26
                                                                                                             91
                                                                                                                   82
                                                                                                                        -12
      113
                 -9
                           -74
                                                 125
                                                        72
                                                            100
                                                                                       -21
                                                                                                             -72
                                                                                                                 -115
                                                                                                                        109
     125
                     -40
                                     -55
                                           -75
                                                            -21
                                                                  18 -53
                                                                           -59
                                                                                   16 -109
                                                                                             -87
                                                                                                             91
```

A.4 Dimension 350

The message is:

```
-23
    -12
          114
                123
                       49
                           -17
                                  17
                                         3
                                            116
                                                  -93
                                                        54
                                                              27
                                                                   121
                                                                         83
                                                                              123
                                                                                    126
                                                                                          125
                                                                                               -70
                                                                                                     -25
                                                                                                                 80
                                                                                                                       -8
                                                                                                                          75
                                                                                                                                -112
                                                                                                                                       -18
-46
-76
     -99
                                                                                                     -50
                                                                   -76
      -7
           -83 -128
                        9
                            28
                                 -46
                                        39
                                            -48
                                                   14
                                                        -19
                                                              80
                                                                         40
                                                                              -62
                                                                                     85
                                                                                           21
                                                                                               -91 -121 -123
                                                                                                                124
                                                                                                                      -35 -84
                                                                                                                                 -17
                                                                                                                                        43
-90
      92
                            46
                                              91
                                                                               72
          121
                -98
                                 -45
                                      126
                                                  -45
                                                               -8
                                                                         -81
                                                                                    -76
                                                                                           29
                                                                                                24
                                                                                                                 91
                                                                                                                      -66
                                                                                                                           91
                                                                                                                                -127
                                                                                                                                       -58
                                                                    38
                                                                                                     -23
                                                                                                           -58
     126
                           -49
                                 -76
                                       -54
                                                        79
                                                                                                -53
                                                                                                                       84
                                                                                                                                       -66
-23
56
            -3
                -59
                        9
                            67
                                 -90
                                            -26
                                                  -76
                                                        116
                                                              53
                                                                   -23
                                                                        -66
                                                                              126
                                                                                     60 -115
                                                                                               117
                                                                                                     -29
                                                                                                            92
                                                                                                                 71
                                                                                                                      -36 -78
                                                                                                                                -110
                                                                                                                                       124
    -90
                       82 -112
           81
                 92
                                 -59
                                       -38
                                              13
                                                   10
                                                        100
                                                             -48
                                                                    49
                                                                        -21
                                                                               55
                                                                                    101
                                                                                          -32
                                                                                               -19
                                                                                                     -49
                                                                                                           30
                                                                                                                -21
                                                                                                                     -116
                                                                                                                          44
                                                                                                                                 -32
                                                                                                                                       -17
     -88
                                                                              -72
           76
                 59
                      -69
                                 -13
                                       -30
                                              26
                                                  -49
                                                                         16
                                                                                          -12
                                                                                                            96
                                                                                                                       55
95 -104 -109
                 11
                       -7
                                        33
                                             -49
                                                   80
                                                         93
                                                                               51
                                                                                     20
                                                                                                -34
                                                                                                     -87 -123
                                                                                                                      -54
                                                                                                                                 -98
                                -108
                                                            -104
                                                                   -60
                                                                                                                                      -101
-39
     -34
           83
                 26
                    -117
                            90
                                  -7
                                       -93
                                            110 -124
                                                         28
                                                             -15
                                                                  -103
                                                                        -66
                                                                               65
                                                                                    105
                                                                                        -100
                                                                                                89
                                                                                                     -83
                                                                                                           -37
                                                                                                                 13
                                                                                                                       66
                                                                                                                                 107
                                                                                                                                       107
                                                       -119
                                                                                                -32
86
           -75
                -49
                            -47
                                  41
                                                                                                                -95
                                                                                                                       69
                       32
                                      126
                                             -91
                                                  -61
                                                            -128
                                                                    60
                                                                         -83
                                                                              110
                                                                                     65
                                                                                          73
                                                                                                      91
                                                                                                         -120
                                                                                                                                       115
                            32
                                                                                                                -67
                -47
                      -12
                                 -89
                                                                                    -28
29
      63
                 66
                       59
                           -10
                                -62
                                      -60
                                             -84
                                                   19
                                                              77
                                                                          47
                                                                               35
                                                                                    -78
                                                                                          -59
                                                                                                76
                                                                                                     -46
                                                                                                           90 -102
                                                                                                                                        88
```

A.5 Dimension 400

The message modulo 6 is:

B Proofs for Section 4

B.1 Proof of Theorem 2

We notice that it suffices to prove the statement for prime powers, by Chinese remainders. Thus, let N be a prime power q^{α} . Since numbers not coprime to N are exactly multiples of q, singular matrices are the matrices for which the determinant is divisible by q. It follows that the proportion of invertible matrices in $\mathcal{M}_n(\mathbb{Z}_{q^{\alpha}})$ is exactly the proportion of invertible matrices in $\mathcal{M}_n(\mathbb{Z}_q)$. Denote by \mathbb{F}_q the finite field with q elements. The number of invertible matrices in $\mathcal{M}_n(\mathbb{F}_q)$ is equal to the number of families $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of linearly independent vectors. A simple counting argument shows that this number is equal to:

$$\prod_{k=0}^{n-1} (q^n - q^k) = q^{n^2} \prod_{k=1}^{n} (1 - q^{-k})$$

B.2 Proof of Theorem 3

We count the number of matrices with one-dimensional kernel by the number of families $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of rank n-1. Denote by B_k the subspace spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_k$, with the convention B_0 being the nullspace. Recall that a k-dimensional subspace has cardinality q^k . For each family $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of rank n-1, there exists a unique i such that $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$ are linearly independent, $\mathbf{b}_i \in B_{i-1}$, and for all j > i, $\mathbf{b}_j \notin B_{j-1}$. There are $\prod_{k=0}^{i-2} (q^n - q^k)$ possibilities for $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$. There are q^{i-1} choices for \mathbf{b}_i . And there are $\prod_{k=i-1}^{n-2} (q^n - q^k)$ possibilities for $(\mathbf{b}_{i+1}, \ldots, \mathbf{b}_n)$. It follows that the total number of families is:

$$\sum_{i=1}^{n} q^{i-1} \prod_{k=0}^{n-2} (q^n - q^k) = \frac{(q^n - 1)q^{n^2}}{(q-1)(q^n - q^{n-1})} \prod_{k=1}^{n} (1 - q^{-k}).$$

Now, consider a family $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of rank n-2. There exists a unique (i_1, i_2) with $i_1 < i_2$ such that: $\mathbf{b}_1, \ldots, \mathbf{b}_{i_1-1}$ are linearly independent, $\mathbf{b}_{i_1} \in B_{i_1-1}$, for all $i_1 < j < i_2, \mathbf{b}_j \notin B_{j-1}$, $\mathbf{b}_{i_2} \in B_{i_2-1}$, and for all $j > i_2, \mathbf{b}_j \notin B_{j-1}$. That way, we know the dimension of B_j for all j, and therefore, the number of $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ corresponding to a given (i_1, i_2) is:

$$\prod_{k=0}^{i_1-2} (q^n - q^k) \times q^{i_1-1} \times \prod_{k=i_1-1}^{i_2-3} (q^n - q^k) \times q^{i_2-2} \times \prod_{k=i_2-2}^{n-3} (q^n - q^k).$$

It follows that the total number of families of rank n-2 is:

$$\prod_{k=0}^{n-3} (q^n - q^k) \times \sum_{i_1=1}^{n-1} \sum_{i_2=i_1+1}^n q^{i_1+i_2-3}.$$

The double sum is equal to:

$$\sum_{i_1=1}^{n-1} \frac{q^{i_1+n-2}-q^{2i_1-2}}{q-1} = \frac{q^{2n-1}-q^n-q^{n-1}+1}{(q-1)(q^2-1)}.$$

Therefore, the number of families is:

$$\frac{q^{2n-1}-q^n-q^{n-1}+1}{(q^n-q^{n-1})(q^n-q^{n-2})(q-1)(q^2-1)}\prod_{k=0}^{n-1}(q^n-q^k).$$

The result follows after a few simplifications.

Weakness in Quaternion Signatures

Don Coppersmith

IBM Research T.J. Watson Research Center Yorktown Heights, NY 10598, USA

Abstract. This note continues a sequence of attempts to define efficient digital signature schemes based on low-degree polynomials, or to break such schemes. We consider a scheme proposed by Satoh and Araki (1997), which generalizes the Ong-Schnorr-Shamir scheme to the non-commutative ring of quaternions. We give two different ways to break the scheme.

1 Introduction

The present note continues a sequence of attempts to define efficient digital signature schemes based on low-degree polynomials, or to break such schemes.

Ong, Schnorr and Shamir [3] presented a signature scheme based on low-degree polynomials modulo a composite integer n of secret factorization, namely $x^2 + ky^2 \equiv m \pmod{n}$. This scheme was subsequently broken by Pollard and Schnorr [4], who used a method of descent to solve this particular polynomial.

A similar scheme was put forth by Shamir [6] and soon broken by Coppersmith, Stern and Vaudenay [2]. These researchers did not solve for the secret key, but found a polynomial satisfied by that key. By an analogy to Galois theory, they adjoined to \mathbf{Z}/n a formal root of this polynomial, performed calculations in this extension ring, and found that the root itself was not required.

A common problem with low-degree polynomial signature schemes is that each signature reveals a polynomial equation satisfied by the secret key. If one collects enough signatures, one can combine the resulting polynomials to gather information about the secret key. We take this route to analyze the present scheme.

The present paper involves a scheme proposed by Satoh and Araki [5], based on the noncommutative ring of quaternions; this scheme is a generalization of the Ong-Schnorr-Shamir [3] scheme. In our solution, we gather three legitimate signatures on arbitrary messages. Each signature gives an equation satisfied by the secret key τ . Combining the three, we can find some scalar multiple π of τ^{-1} , such that $\pi\tau$ is an unknown square root of a known element of \mathbf{Z}/n . Working in the quaternions we are able to get around this square root, producing a key ν which will work equally as well as τ for signing future messages.

Our paper is organized as follows. In section 2 we review the ring of quaternions, especially as used with the integers mod n. Section 4 describes the Satoh-Araki scheme. In section 5 we show how to collect and solve equations involving

the secret key τ , and produce the equivalent key ν , with which future messages can be signed. A second solution is given in section 6, which does not need to see legitimate signatures, but which requires a bit of computation to produce each new signature. Section 7 demonstrates that we cannot push these attacks further; we cannot obtain the secret key, either for this scheme or the original Ong-Schnorr-Shamir scheme. We conclude in section 8.

2 Quaternions Modulo n

The Satoh-Araki signature scheme operates in a ring R of quaternions modulo a composite integer n. The factorization of n is secret. Even the legitimate user need not know the factorization.

An element α of the ring R is a 4-tuple (a, b, c, d) of elements of \mathbf{Z}/n (the integers modulo n). This element is usually written as $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$. The special elements $\mathbf{i}, \mathbf{j}, \mathbf{k}$ satisfy the non-commutative multiplication rules:

$$i^{2} = j^{2} = k^{2} = -1$$

 $i j = k = -j i$
 $j k = i = -k j$
 $k i = j = -i k$

Greek letters $\alpha, \beta...$ represent elements of R, while Roman letters a, b, c, ... represent elements of \mathbf{Z}/n . We denote by α^* the Hermite conjugate of α :

$$(a, b, c, d)^* = (a, -b, -c, -d);$$

by $N(\alpha)$ the norm of α :

$$N(a, b, c, d) = (a, b, c, d)(a, b, c, d)^* = a^2 + b^2 + c^2 + d^2 \in \mathbf{Z}/n;$$

and by α^T the *transpose* of α :

$$(a, b, c, d)^T = (a, b, -c, d).$$

Elements of the form (a, b, 0, d) are termed *symmetric* because they satisfy $\alpha = \alpha^T$. Elements of the form $(a, 0, 0, 0) \in \mathbf{Z}/n$ are called *scalars*.

Multiplication is non-commutative.

The multiplicative group of invertible elements of R is denoted R^{\times} . The inverse is computed by

$$\alpha^{-1} = (\alpha^* \alpha)^{-1} \alpha^* = N(\alpha)^{-1} \alpha^*$$

whenever it exists, that is, whenever $N(\alpha)$ is relatively prime to n; recall that $N(\alpha)$ is a scalar so that its inversion is easy.

The transpose satisfies $(\alpha \beta)^T = \beta^T \alpha^T$. We also have $(\alpha^T)^{-1} = (\alpha^{-1})^T$.

The powers of any element α are integer linear combinations of 1 and α . In particular, if $\alpha = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$,

$$\alpha^2 = \alpha(2a - \alpha^*) = -N(\alpha) + 2a\alpha,$$

so that α^2 is a linear combination of 1 and α , and the other powers follow by induction.

3 The Pollard-Schnorr Result

We will use the result due to Pollard and Schnorr [4]:

Theorem 1. Suppose the Generalized Riemann Hypothesis holds. Then there is a probabilistic algorithm which, upon input k, m and n with gcd(km, n) = 1, will solve $x^2 + ky^2 \equiv m \pmod{n}$ with an expected number of $O((\log n)^2 |\log \log |k||)$ arithmetical operations on $O(\log n)$ -bit numbers.

We will also use a generalization due to Adleman, Estes and McCurley [1]:

Theorem 2. Let n be an odd positive integer, and let f(x,y) be given by $f(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F$, and define $\Delta(f)$, the determinant of f, as follows:

$$\Delta f = \det \begin{bmatrix} 2A & B & D \\ B & 2C & E \\ D & E & 2F \end{bmatrix}.$$

If $\gcd(\Delta f, n) = 1$, then there exists an algorithm requiring $O(\log(\epsilon^{-1}\log n)\log^4 n)$ arithmetic operations on integers of size $O(\log n)$ bits that will give a solution to $f(x, y) \equiv 0 \pmod{n}$ with probability $1 - \epsilon$.

Remark: The generalization of Theorem 1 to general quadratic forms (Theorem 2) is achieved by completing the square, but Theorem 2 also dispenses with the Generalized Riemann Hypothesis.

4 The Satoh-Araki Scheme

The Satoh-Araki scheme generalizes the Ong-Schnorr-Shamir scheme to the ring of quaternions mod n. In this scheme n is a large composite modulus whose factorization is not public; even the legitimate user need not know the factorization. The secret key is a random ring element $\tau \in R^{\times}$. The public key κ is the symmetric ring element

$$\kappa = -(\tau^T)^{-1}\tau^{-1}.\tag{1}$$

A message μ is encoded as a *symmetric* element of R. A signature (σ_1, σ_2) of μ is computed as follows: Pick $\rho \in R^{\times}$ randomly. Compute

$$\sigma_1 = \rho^{-1}\mu + \rho^T, \quad \sigma_2 = \tau(\rho^{-1}\mu - \rho^T).$$

One verifies the signature by checking the equation

$$4\mu = \sigma_1^T \sigma_1 + \sigma_2^T \kappa \sigma_2.$$

5 Breaking the Scheme

For our first solution we are going to produce an "equivalent" secret key ν .

Definition 3. Two secret keys τ, ν in R^{\times} are equivalent if they correspond to the same public key κ by equation (1); that is, they satisfy

$$\kappa = -(\tau^T)^{-1}\tau^{-1} = -(\nu^T)^{-1}\nu^{-1},$$

which is to say

$$-1 = \tau^T \kappa \tau = \nu^T \kappa \nu.$$

As a first step towards this goal, we need to see the signatures of three arbitrary messages. Each such signature (σ_1, σ_2) satisfies the important property that

$$\sigma_1^T \tau^{-1} \sigma_2$$
 is symmetric. (2)

We verify this as follows:

$$\begin{split} \sigma_1^T \tau^{-1} \sigma_2 &= (\rho^{-1} \mu + \rho^T)^T (\tau^{-1}) (\tau(\rho^{-1} \mu - \rho^T)) \\ &= (\mu(\rho^T)^{-1} + \rho) (\rho^{-1} \mu - \rho^T) \\ &= \mu(\rho^{-1})^T \rho^{-1} \mu - \mu(\rho^T)^{-1} \rho^T + \rho \rho^{-1} \mu - \rho \rho^T \\ &= \mu(\rho^{-1})^T \rho^{-1} \mu - \mu + \mu - \rho \rho^T \\ &= \mu(\rho^{-1})^T \rho^{-1} \mu - \rho \rho^T \end{split}$$

and each term is manifestly symmetric. (Recall that μ is symmetric.)

We would like to calculate τ or τ^{-1} , but this is too hard. (See section 7.) Instead, we will find an element π which is a scalar multiple of τ^{-1} . Each such scalar multiple $\pi = \ell \tau^{-1}, \ell \in \mathbf{Z}/n$, also satisfies the property that $\sigma_1^T \pi \sigma_2$ is symmetric. This is a linear homogeneous condition on the coefficients of π .

Suppose we see three signatures $(\sigma_1^{(i)}, \sigma_2^{(i)})$ on three messages $\mu^{(i)}$, i = 1, 2, 3, all produced by the same secret key τ . Each signature gives a linear homogeneous relation on the coefficients of π , namely that $(\sigma_1^{(i)})^T \pi \sigma_2^{(i)}$ is symmetric. By experiment we see that these three relations are in general nonredundant.

Remark: The three equations being redundant would correspond to the vanishing of a certain 3×3 determinant modulo p, where p is one of the unknown factors of the integer n. This determinant is a polynomial of low degree d in several random variables. The fact that the determinant failed to vanish in our experiments, implies that the determinant is not identically $0 \pmod{p}$, which implies that its probability of vanishing is O(d/p). Since p is so large we can safely ignore this probability of failure. Even in the remote case of failure, if a determinant vanished modulo p but not modulo p, then the Euclidean algorithm would allow us to factor p via $\gcd(\det p) = p$. A similar situation will hold whenever we "hope" that something does not "accidentally" vanish.

Since the three relations are nonredundant, they restrict the space of possible π to a one-dimensional space. That is, they determine π up to an unknown multiplicative scalar ℓ : $\pi = \ell \tau^{-1}, \ell \in \mathbf{Z}/n$. We select one such representative π .

We know the public key $\kappa = -(\tau^T)^{-1}\tau$. So we can compute

$$\begin{array}{l} z = (\pi^T)^{-1} \kappa \pi^{-1} \\ = \ell^{-1} \tau^T \left(-(\tau^T)^{-1} \tau^{-1} \right) \tau \ell^{-1} \\ = -\ell^{-2} \in \mathbf{Z}/n \end{array}$$

We know z but not ℓ . It is infeasible to take square roots in \mathbb{Z}/n , so that we cannot compute ℓ from z. But in the quaternions we can easily find an element with a given norm, and this will serve in place of finding a square root.

Here we use a special case of the Pollard-Schnorr attack (Theorem 1) where k=1, to find integers c,d satisfying

$$c^2 + d^2 \equiv -z^{-1} \pmod{n}.$$

Then

$$(c+d\mathbf{j})^T(c+d\mathbf{j}) = (c-d\mathbf{j})(c+d\mathbf{j}) = c^2 + d^2 \equiv -z^{-1} \pmod{n}.$$

We can now define our "equivalent key" ν :

$$\nu = \pi^{-1}(c + d\mathbf{j}).$$

The equation (1) relating κ and τ can be restated as

$$\kappa = -(\tau^T)^{-1}\tau^{-1}$$
$$-1 = \tau^T \kappa \tau$$

We show that this equation is also satisfied by ν in place of τ :

$$\begin{split} \boldsymbol{\nu}^T \kappa \boldsymbol{\nu} &= (c+d\mathbf{j})^T (\pi^{-1})^T \kappa \pi^{-1} (c+d\mathbf{j}) \\ &= (c-d\mathbf{j}) \left((\pi^{-1})^T \kappa \pi^{-1} \right) (c+d\mathbf{j}) \\ &= (c-d\mathbf{j}) z (c+d\mathbf{j}) \\ &= (c-d\mathbf{j}) (c+d\mathbf{j}) z \\ &= (c^2+d^2) z \\ &\equiv -1 \pmod{n}. \end{split}$$

Thus the "secret key" ν corresponds to the public key κ in the prescribed manner; ν and τ are "equivalent" by our definition. This implies that the attacker can use ν to create signatures, exactly as the legitimate user uses τ .

To compute ν we only needed to see three legitimate signatures and do a minimal amount of computation.

In some sense this attack is unsatisfactory. It depended on property (2), which in turn depended on the very structured way that σ_1, σ_2 were computed. They could have been computed in a more random fashion; for example, σ_1 could have been left-multiplied by a random element β satisfying $\beta^T \beta = 1$, freshly calculated for each message, which would not affect the validity of the signature, but would block the present attack. So in the next section we present an attack that does not depend on the particular method of generating signatures outlined in [5].

6 A Second Attack

In our second attack, we do not need to see any legitimate signatures. We need only the public key κ and modulus n. To sign a given message μ , we will perform three Pollard-Schnorr computations.

We are given the public key κ and a message μ , both symmetric elements of R, and we are required to find elements σ_1, σ_2 of R satisfying $\sigma_1^T \sigma_1 + \sigma_2^T \kappa \sigma_2 = 4\mu$.

The space of symmetric elements of R is a three-dimensional linear space over \mathbf{Z}/n . With very high probability the three symmetric elements 1, κ , μ form a linear basis for this space; we assume this to be the case.

For unknown elements a, b, d of \mathbf{Z}/n , consider the product $S = (a + b\mathbf{i} + d\mathbf{k})^T \kappa (a + b\mathbf{i} + d\mathbf{k})$. Being symmetric, S can be expressed as a linear combination of 1, \mathbf{i} and \mathbf{k} , with coefficients being quadratic functions of a, b, d. That is,

$$(a + b\mathbf{i} + d\mathbf{k})^{T} \kappa(a + b\mathbf{i} + d\mathbf{k}) = Q_{1}(a, b, d)1 + Q_{2}(a, b, d)\mathbf{i} + Q_{3}(a, b, d)\mathbf{k},$$
$$Q_{i}(a, b, d) = q_{i11}a^{2} + q_{i12}ab + q_{i13}ad + q_{i22}b^{2} + q_{i23}bd + q_{i33}d^{2},$$
$$q_{ijk} \in \mathbf{Z}/n.$$

The entries q_{ijk} of Q_i are linear functions of the entries of κ .

A preview of the computation: We will find a setting of a,b,d making S be a linear combination of 1 and μ . This enables us to arrange that in our signature equation $4\mu \stackrel{?}{=} \sigma_1^T \sigma_1 + \sigma_2^T \kappa \sigma_2$, both sides lie in the two-dimensional subspace spanned by 1 and μ . We can select parameters to make the coefficients of μ agree, and then the coefficients of 1, so that the signature equation will hold. At each stage we will need to solve a Pollard-Schnorr equation.

Let $\mu = m_1 + m_2 \mathbf{i} + m_3 \mathbf{k}$ with $(m_2, m_3) \neq (0, 0)$, and set

$$R(a,b,d) = \det \begin{bmatrix} 1 & 0 & 0 \\ m_1 & m_2 & m_3 \\ Q_1(a,b,d) \; Q_2(a,b,d) \; Q_3(a,b,d) \end{bmatrix},$$

$$R(a, b, d) = m_2 Q_3(a, b, d) - m_3 Q_2(a, b, d).$$

R(a,b,d) is a quadratic function of a,b,d. Our first task is to find a,b,d (not all zero) such that $R(a,b,d)\equiv 0\pmod n$; this is equivalent to S being a linear combination of 1 and μ . For this purpose we use Theorem 2, with d=1, $a=x,\,b=y$ and R(a,b,1)=f(x,y). For this theorem we need to assume that $\gcd(\Delta(f),n)=1$, that is, that for each prime p dividing $n,\,\Delta f\neq 0\pmod p$. But each coefficient of R(a,b,1) is a polynomial of total degree 2 in the coefficients of μ and κ , so that Δf is a polynomial of total degree 6 in the coefficients of μ and κ . Also, Δf is not identically 0 (because it is nonzero in some experimental instances), so it will be 0 $\pmod p$ with negligible probability O(1/p), with probability being taken over random problem instances (κ,μ) and for p sufficiently large. So with high probability Theorem 2 applies, and we can easily find a,b satisfying $R(a,b,1)\equiv 0\pmod n$.

This means that we have computed scalars a, b, c, e satisfying

$$(a+b\mathbf{i}+\mathbf{k})^T\kappa(a+b\mathbf{i}+\mathbf{k})=c+e\mu.$$

For scalars f, g, h, w yet undetermined, we are going to have

$$\sigma_1 = h + w\mathbf{j}$$

 $\sigma_2 = (a + b\mathbf{i} + \mathbf{k})(f + g\mu)$

Then our desired signature equation will be

$$4\mu \stackrel{?}{=} \sigma_1^T \sigma_1 + \sigma_2^T \kappa \sigma_2$$

$$= (h + w\mathbf{j})^T (h + w\mathbf{j}) + (f + g\mu)^T (a + b\mathbf{i} + \mathbf{k})^T \kappa (a + b\mathbf{i} + \mathbf{k}) (f + g\mu)$$

$$= (h^2 + w^2) + (f + g\mu)(c + e\mu)(f + g\mu)$$

$$= (h^2 + w^2 + cf^2) + (2cfg + ef^2)\mu + (2efg + cg^2)\mu^2 + (eg^2)\mu^3$$

As noted in Section 2, μ^2 and μ^3 are linear combinations of μ and 1. Suppose we calculate

$$\mu^{2} = z\mu + r$$
$$\mu^{3} = s\mu + t$$
$$z, r, s, t \in \mathbf{Z}/n$$

Then our desired equation is

$$4\mu \stackrel{?}{=} (h^2 + w^2 + cf^2 + r(2efg + cg^2) + t(eg^2)) + [2cfg + ef^2 + z(2efg + cg^2) + s(eg^2)]\mu$$

The free variables are f, g, h, w, and the known constants are c, e, r, s, t, z, and the ring element μ .

The coefficient of μ in the above equation is a quadratic in f, g. We use Theorem 2 to find f, g satisfying

$$4 = 2cfq + ef^{2} + z(2efq + cq^{2}) + s(eq^{2}).$$

Having done this, another application recovers unknowns h, w satisfying

$$0 = (h^2 + w^2) + cf^2 + r(2efq + cq^2) + t(eq^2).$$

Putting it all together, we have used Pollard-Schnorr or its generalization (Adleman-Estes-McCurley) three times to find a signature (σ_1, σ_2) satisfying the signature equation for a given κ, μ .

Remark: The Pollard-Schnorr solution to the equation $x^2 + ky^2 \equiv m \pmod{n}$ requires that both k and m be relatively prime to n. In each of our applications of the solution, this will be the case with high probability.

7 Impossibility Results

We collect here some impossibility results, showing that in some sense our attacks are the best possible.

In our first attack, we found a scalar multiple of the secret key τ . We also found an "equivalent" secret key ν which we could use in place of τ to sign messages. But it is infeasible to find an equivalent secret key which is simultaneously a scalar multiple of the true secret key, even given the signatures of many chosen messages. The same is true of the original Ong-Schnorr-Shamir scheme.

In our second attack, knowing only the public key, we can generate valid signatures of arbitrary messages. But without seeing signatures generated by the legitimate owner, it is infeasible to compute an equivalent secret key.

Theorem 4. Assume it is infeasible to factor n. Then, given the legitimate signatures of polynomially many chosen messages, it is infeasible to find any quantity ν which is both a scalar multiple of the secret key τ and also an equivalent secret key.

Proof. The legitimate secret key τ and nonce ρ generate a signature (σ_1, σ_2) on the message μ by

 $\sigma_1 = \rho^{-1}\mu + \rho^T$ $\sigma_2 = \tau(\rho^{-1}\mu - \rho^T).$

Using the same process, an alternate secret key $\tau' = -\tau$ and nonce $\rho' = \mu(\rho^{-1})^T$ would generate a signature (σ'_1, σ'_2) on the same message by

$$\sigma_1' = \rho^T \mu^{-1} \mu + \rho^{-1} \mu = \sigma_1
\sigma_2' = -\tau (\rho^T \mu^{-1} \mu - \rho^{-1} \mu) = \sigma_2.$$

So, with arbitrary chosen plaintext, we cannot distinguish between the secret keys τ and τ' .

Suppose (without loss of generality) that n = pq is the product of two primes. Consider a third secret key τ'' , satisfying

$$\tau'' \equiv \left\{ \begin{array}{ll} \tau \pmod{p} \\ \tau' \pmod{q} \end{array} \right\}$$

By the Chinese Remainder Theorem, τ'' would also be an acceptable secret key. It happens that the only "equivalent keys" which are scalar multiples of τ are $\pm \tau$ and $\pm \tau''$. That is, if $\nu = \ell \tau$ is a scalar multiple of τ satisfying

$$\kappa = -(\tau^T)^{-1}\tau^{-1} = -(\nu^T)^{-1}\nu^{-1}$$

then we necessarily have

$$\ell^2 = 1 \in \mathbf{Z}/n,$$

and the four roots of this equation correspond to $\pm \tau$ and $\pm \tau''$.

Suppose we have an oracle capable of recovering an equivalent secret key which is simultaneously a scalar multiple of the true secret key, using the signatures of polynomially many chosen messages. Then we can factor n. Namely, given n, we select τ randomly, compute the corresponding public key κ , and begin producing signatures. (Recall that we do not need to know the factorization of n to do so.) Using the oracle, we recover a key, either $\pm \tau$ or $\pm \tau''$. The

recovered key will be unequal to $\pm \tau$ with probability at least 1/2; say the key is τ'' . Each coordinate of $\tau'' - \tau$ is divisible by p, and at least one coordinate is not divisible by q, so that for the price of computing a few g.c.d.s with n we will recover p.

Remark: The same idea shows that in the original Ong-Schnorr-Shamir scheme, even with polynomially many signatures of chosen messages, it is infeasible to recover an "equivalent secret key", namely a square root of the public key.

Our next result shows that, if we have no legitimate signatures, the second attack is the best we can hope for.

Theorem 5. Given only the public key, we cannot find an equivalent secret key.

Proof. An oracle to do so would enable us to factor n. Namely, select a random integer x and compute $z \equiv x^2 \pmod{n}$. Use Pollard-Schnorr to find integers c, d satisfying $c^2 + d^2 \equiv z \pmod{n}$. Define a public key $\kappa = c + d\mathbf{i}$. Use the oracle to find a ring element τ satisfying $\tau^T \kappa \tau = -1$. By multiplicativity of norm, we know then $N(\tau^T)N(\kappa)N(\tau) = N(-1)$. But $N(\kappa) = c^2 + d^2 \equiv x^2 \pmod{n}$, whence $1 = N(\tau)^2N(\kappa) = (N(u)x)^2 \pmod{n}$, so that $\gcd(n, N(u)x - 1)$ is (with probability at least 1/2) a nontrivial factor of n.

8 Conclusions

We have presented two solutions to the Satoh-Araki signature scheme. The first depended on the particular way of generating signatures outlined in [5] to generate linear equations on the coefficients of the secret key τ , giving us an unknown scalar multiple of τ , related by a square root. We finessed the square root calculation by taking advantage of the freedom of the quaternion ring. The second solution worked only from the public key and the message, with no need to see previous legitimate signatures, and worked with high probability, requiring only three applications of a Pollard-Schnorr solution. Both are computationally quite efficient.

References

- 1. L.M. Adleman, D.R. Estes and K.S. McCurley, "Solving bivariate quadratic congruences in random polynomial time," *Math. Comp.* 48 (1987), pp. 17–28.
- D. Coppersmith, J. Stern and S. Vaudenay, "Attacks on the birational permutation signature schemes," Advances in Cryptology - CRYPTO '93, Springer LNCS 773, D.R. Stinson (Ed.), pp. 435-443. Journal of Cryptology, Vol. 10, No. 3 (Summer 1997), pp. 207-221.
- 3. H. Ong, C.P. Schnorr and A. Shamir, "An efficient signature scheme based on quadratic equations," *Proc. 16th ACM Symp. Theory of Computation*, 1984, pp. 208–216.
- 4. J.M. Pollard and C.P. Schnorr, "An efficient solution of the congruence $x^2 + ky^2 \equiv m \pmod{n}$," *IEEE Trans. Inf. Theory*, Vol. IT-33, 1987, pp. 702–709.

314 Don Coppersmith

- 5. T. Satoh and K. Araki, "On construction of signature scheme over a certain non-commutative ring," *IEICE Trans. Fundamentals*, Vol. E80-A, No. 1, January 1997, pp. 40-45.
- 6. A. Shamir, "Efficient signature schemes based on birational permutations," Advances in Cryptology CRYPTO '93, Springer LNCS **773**, D.R. Stinson (Ed.), pp. 1–12.

Cryptanalysis of "2R" Schemes

Ye Ding-Feng¹, Lam Kwok-Yan², and Dai Zong-Duo³

Kent Ridge Digital Lab, Singapore 119613
 dfye@krdl.org.sg
 National University of Singapore, Singapore 119260
 lamky@comp.nus.edu.sg
 Graduate School, University of Science & Technology of China
 yangdai@mimi.cnc.ac.cn

Abstract. The function decomposition problem can be stated as: Given the algebraic expression of the composition of two mappings, how can we identify the two factors? This problem is believed to be in general intractable [1]. Based on this belief, J. Patarin and L. Goubin designed a new family of candidates for public key cryptography, the so called "2R-schemes" [10, 11]. The public key of a "2R"-scheme is a composition of two quadratic mappings, which is given by n polynomials in n variables over a finite field K with q elements. In this paper, we contend that a composition of two quadratic mappings can be decomposed in most cases as long as q > 4. Our method is based on heuristic arguments rather than rigorous proofs. However, through computer experiments, we have observed its effectiveness when applied to the example scheme " D^{***} " given in [10].

1 Introduction

Public key cryptography is becoming more and more important in modern computer and communication systems. Many public key cryptosystems (PKCs) have been proposed since Diffie and Hellman initiated this direction in 1976 [2]. Usually the security of a PKC relies on a hard mathematical problem. The most famous such problems are integer factorization and discrete logarithm. PKCs based on these two kinds of problems, such as RSA[13] and ElGamal[3], although mathematically sound, need to perform a large amount of huge arithmetics, so are not very efficient compared to classical symmetric cryptographic algorithms such as DES. Much effort has been paid in seeking more efficient constructions for PKCs. One class of these constructions make use of mapping compositions. The basic idea is as follows: a user chooses several easily-invertible mappings which he keeps secret, computes the algebraic expression of their composition and makes it public; then anyone else can do encryption or verify signatures using the public key, but will be faced with a set of complicated algebraic equations when he tries to decrypt cipher texts or to forge signatures. An obvious advantage of these PKCs is that the private key side computations (decrypting and signing) can be made very efficient and be implemented with very simple

hardware. There are two main drawbacks however: large public key size and ambiguous security foundations.

The earliest examples of PKCs making use of mapping compositions were proposed by T.Matsumoto and H.Imai [4] in 1985. One of them, called "B", looks like " $t \circ f \circ s$ ", where t, s are two secret linear mappings over $GF(2)^n$, $f: x \mapsto (x+c) \mod (2^n-1)+1$, $0 \mapsto 0$, c is also secret, and elements of $GF(2)^n$ is identified with integers naturally. This scheme is still unbroken. Another example is called " C^* " (see also [5]), in which the above f is replaced by a "quadratic polynomial tuple" which will be called quadratic mapping in this paper. C^* was broken by Jacques Patarin [7] in 1995.

One-round schemes are generalizations of C^* . They are of the form the " $t \circ f \circ s$ ", where $s, t: K^n \to K^n$ are affine, $f: K^n \to K^n$ is quadratic, and K is a finite field. J. Patarin and L. Goubin gave several constructions of one-round schemes using algebraic techniques and S-boxes (see [10,11]), and they also showed that their constructions are insecure. Therefore they proposed two-rounds schemes, abbreviated as "2R", in which the public key is the composition of two secret one-round schemes, based on the assumption that functional decomposition problem is hard.

In this paper, we show that "2R" schemes can be decomposed into separated one-round schemes in most cases as long as the field K has more than 4 elements. However, we were only able to justify this claim by some heuristic arguments and experimental evidences instead of rigorous proofs.

Briefly stated, our method is as follows. Suppose $\pi = f \circ g: K^n \to K^n$ be the composition of two quadratic mappings. We have n output polynomials of π in n variables of degree 4. The partial derivatives of all these polynomials with respect to all the n input variables give n^2 cubic polynomials, spanning a linear space \tilde{V} . This space is contained in the space V of cubic polynomials spanned by products of the n input variables X_i and the n intermediate output polynomials of g, provided that K has more than 4 elements. Since both \tilde{V} and V tend to have dimension n^2 for random choices, we hope they are equal (or at least the the codimension is small). For a linear combination F of input variables, we can use linear algebra to compute (V:F), the space of quadratic polynomials r such that $rF \in V$. When n > 2, the intersection of these spaces is a candidate for the space $\mathcal{L}(g)$ spanned by the n output polynomials of g. This last statement needs the assumption that the factorization of π is unique, that is, if we write $\pi = f' \circ g'$ for quadratic f', g', then g and g' differ only by a linear factor.

We have applied this method to a concrete example D^{**} in the "2R" family. D^{**} is a composition of two D^{*} s, and a D^{*} is a mapping of the form $t \circ \phi \circ s$ ", where ϕ is the squaring in the extension field $K^{(n)}$. In the example, $K = \mathrm{GF}(251)$ and n = 9. In our experiments, the above method has never failed to find the linear class of the inner D^{*} , by which we mean the set of mappings which differ from each other by a linear bijection.

The rest of this paper is organized as follows: Section 2 gives a brief review of "2R" schemes and some notations and definitions. Section 3 describes the

steps in decomposing compositions of quadratic mappings. Section 4 gives some experiment reports. Section 5 is the conclusion of this paper.

2 "2R" Schemes and " D^{**} ": A Brief Review

Through out this paper K denotes a finite field of q elements, and K^n denotes the vector space over K of dimension n. Any polynomial $P = P(X_1, X_2, \dots, X_n)$ can be seen as a mapping $K^n \to K: (x_1, x_2, \dots, x_n) \mapsto P(x_1, x_2, \dots, x_n)$. Similarly, any n polynomials (P_1, P_2, \dots, P_n) can be regarded as a mapping $K^n \to K^n$:

$$(x_1, x_2, \dots, x_n) \mapsto (P_1(x_1, x_2, \dots, x_n), P_2(x_1, x_2, \dots, x_n), \dots, P_n(x_1, x_2, \dots, x_n).$$

Conversely, any mapping $K^n \to K^n$ can be expressed as n polynomials as above, these polynomials are called its component polynomials. A mapping is called *linear*, if its component polynomials are all homogeneous of degree 1; affine, if constant terms are allowed; quadratic, if the total degree ≤ 2 .

"2R" schemes ("2R" stands for 2 rounds) were introduced by Jacques Patarin and Louis Goubin in [11]. The private key consists of

- 1. Three affine bijections r, s, t from K^n to K^n .
- 2. Two quadratic mappings ψ , $\phi:K^n\to K^n$ (in fact, these two mappings can also be made public).

The public items are:

- 1. The field K and dimension n.
- 2. The *n* polynomials of the composed mapping $\pi = t \circ \psi \circ s \circ \phi \circ r$ which are of total degree 4.

The public-key side computation is just an application of the mapping π (both message blocks and signatures belong to K^n). To explain decryption and signing, we need more words. The designers of these schemes do not require the private mappings ψ , ϕ be bijections. To achieve the uniqueness of decryption, we should introduce enough redundancy in message blocks. Similarly, to compute a signature, we should keep enough redundancy-bits so that for any message m, we can find a redundant tail R making m||R lie in the range of π . The non-injectiveness of ψ , ϕ will in general greatly reduce the efficiency in private-key side computations. In the scheme D^{**} , these drawbacks are overcomed by a clever choice of the message-block space, see [10]. The essence of decryption is to find the full preimage $\pi^{-1}(c)$ for any given c, and that of signing is to find a single element belonging to $\pi^{-1}(c)$. When the private keys are known, this can be reduced to inverting ψ and ϕ .

As the authors of [11] point out, the security of "2R" schemes can be affected by the choices of ψ , ϕ . Since ψ and ϕ should be easy to construct and invert, currently only the following constructions are known:

1. "C*-functions": monomials over an extension of degree n over $K: a \mapsto a^{1+q^{\theta}}$.

2. "Triangular-functions":

$$(a_1, \dots, a_n) \mapsto (a_1, a_2 + q_1(a_1), \dots, a_n + q_{n-1}(a_1, \dots, a_{n-1}))$$

where each q_i is quadratic.

3. "S-boxes-functions": $(a_1, \dots, a_n) \mapsto$

$$(S_1(a_1, \dots a_{n_1}), S_2(a_{n_1+1}, \dots, a_{n_1+n_2}), \dots, S_d(a_{n_1+\dots+n_{d-1}}, \dots, a_n))$$

where $n = \sum n_i$, and S_i is a quadratic mapping $K^{n_i} \to K^{n_i}$.

- 4. techniques by combining "S-boxes" with "triangular-functions".
- 5. D^* -functions: squaring in extension of K of degree n, denoted as $K^{(n)}$, where $q^n \equiv -1 \pmod{4}$.

Previous researches [11, 8] have shown that, when ψ is in the first two classes, the resulted scheme is weak. Note that if we drop t and ψ in above description of "2R", we get the so called one-round schemes. A "2R" scheme is just a composition of two one-round schemes. All one-round schemes from the above constructions have been shown to be insecure [8, 9, 7, 10, 11].

" D^{**} " is a special instance of "2R". It is defined as:

- 1. $q^n \equiv -1 \pmod{4}$, and q is about of the size 2^8 . (For example, q = 251, $n = 9 \lceil 10 \rceil$.)
- 2. r, s, t are linear bijections.
- 3. $\psi = \phi$ is the squaring in $K^{(n)}$, where $K^{(n)}$ denotes the extension of K of degree n.
- 4. The message block space is chosen in such a way [10] that the restriction of π on it is an injection. (This is irrelevant to the purpose of this paper.)

Note that the public polynomials in D^{**} are all homogeneous of degree 4.

3 Decomposing "2R" Schemes

A basic assumption behind "2R" schemes is that the functional decomposition problem for a composition of two quadratic mappings from K^n to K^n is hard. In this section we will give evidences which indicate that this assumption is not realistic provided q > 4.

As in the previous section, let $\pi = t \circ \psi \circ s \circ \phi \circ r$ be the public key. If for any quadratic f, g, satisfying $\pi = f \circ g$, we have $f = t \circ \psi \circ s_1$, $g = s_2 \circ \phi \circ r$, for some affine bijections s_1 , s_2 satisfying $s = s_1 \circ s_2$, we say that π has unique factorization. If the factorization of π is not unique, even we can decompose it into two quadratic mappings, we are not sure if these two mappings are one-round functions which can be attacked by known methods. Therefore we need to assume this uniqueness of decomposition. It seems difficult to justify this assumption theoretically, but we believe that most compositions of quadratic mappings do have unique factorizations.

Note that if we can find a $g = s_2 \circ \phi \circ r$, then f may be obtained by solving linear equations arising from coefficients-comparing. Note also that s_2 is not important, what we really care is the affine class $\{s \circ \phi \circ r : \text{for all affine bijection } s\}$ (similarly, the linear class of a mapping g is $\{s \circ g : \text{for all linear bijection } s\}$), and this class is uniquely determined by the vector space generated by component polynomials of g and 1. In the following we will describe how to obtain this space when given the component polynomials of π .

To ease the discussion, we assume all the mappings r, s, t, ϕ , ψ are homogeneous. In this case, we only need q > 3. The general case can be reduced to the homogeneous case when q > 4, by a standard algebraic procedure which is called homogenization, see Appendix 1.

3.1 A Linear-Algebra Problem on Polynomials

Now we assume f, g be two homogeneous quadratic mappings from K^n to K^n . Given the composition $f \circ g$, which is a homogeneous mapping of degree 4, we want to determine the linear class of g, this is equivalent to determine the linear space $\mathcal{L}(g)$ generated by component polynomials of g. This linear space may not be directly obtained, but later we will show that the linear space $V(g) = \sum_{1 \leq i \leq n} X_i \mathcal{L}(g)$ can in most cases be obtained from the component polynomials of $f \circ g$. So we are faced with the following problem of linear algebra.

Problem 1 Let W be a linear space of dimension $\leq n$ consisting of quadratic forms in n variables X_1, \dots, X_n . Given $V = \sum_{1 \leq i \leq n} X_i W$, is it possible (and how) to uniquely determine W?

For any subspace \mathcal{L}' of the linear space \mathcal{L} generated by X_1, \dots, X_n , let

$$(V:\mathcal{L}') \stackrel{\mathrm{def}}{=} \{r \in K[X_1, X_2, \cdots, X_n]: r\mathcal{L}' \subseteq V\}.$$

When \mathcal{L}' has dimension 1, say, generated by F, we also write $(V:F)=(V:\mathcal{L}')$. We have the following conjecture.

Conjecture 1 Notations and assumptions as above, then for randomly chosen W, the probability ρ that $(V : \mathcal{L}) = W$ are very close to 1 when n > 2.

Note that $(V : \mathcal{L}')$ can be computed using linear algebra for any V and \mathcal{L}' , so the above conjecture says that in general the answer to the above problem is positive.

Although we can not prove the above conjecture or give a reasonable estimation on ρ , in the following we will justify this conjecture with some heuristic arguments based on some standard facts from linear algebra.

Let \mathcal{Q} denote the total space of all quadratic forms. We have $\dim(\mathcal{Q}) = n(n+1)/2$. In the application at hand we may assume $\dim(\mathcal{W}) = n$, so $\dim(\mathcal{Q}/\mathcal{W}) = n(n+1)/2 - n = n(n-1)/2$, where \mathcal{Q}/\mathcal{W} means quotient space. Now we wish to estimate $\dim((V:\mathcal{L})/\mathcal{W})$. Note that $(V:\mathcal{L})/\mathcal{W} = \bigcap_i (V:X_i)/\mathcal{W}$. It is not easy to characterize this intersection because of the complex relations between

320

the spaces $(V:X_i)$. To simplify things, we regard the n spaces $(V:X_i)/\mathcal{W}$ as n independent random variables. This is neither supported or disapproved by any theoretical results we know. By linear algebra (see Appendix 2), two random subspaces of dimension n_1, n_2 of a n-dimension space tend to have intersection of dimension $n_1 + n_2 - n$, so we need that $\sum_i \dim((V:X_i)/\mathcal{W})$ exceeds (n-1)n(n-1)/2 to expect a nonzero intersection $\bigcap_i (V:X_i)/\mathcal{W}$.

Now let us see the dimension of the subspaces $(V:X_i)/\mathcal{W}$. Since every coordinate X_i plays the same role, we only need to consider $(V:X_1)/\mathcal{W}$. Let (g_1, g_2, \dots, g_n) be a basis of \mathcal{W} , any element in $(V:X_1)/\mathcal{W}$ can be written in form $(\sum g_i F_i)/X_1$, where F_i are linear forms in X_2, \dots, X_n , and satisfying $\sum g_i(0, X_2, \dots, X_n)F_i = 0$. Let σ be the linear map from \mathcal{L}'^n , where \mathcal{L}' be the space of linear forms in $X_2, \dots X_n$, to the space of cubic polynomials:

$$(F_1, \dots, F_n) \mapsto \sum g_i(0, X_2, \dots, X_n) F_i.$$

Then we see $\dim((V:X_1)/\mathcal{W}) \leq \dim(\ker(\sigma))$. Again we regard σ as a random linear mapping between spaces of dimensions n(n-1), (n-1)n(n+1)/6 respectively, so we may expect $\dim(\sigma) = n(n-1) - (n-1)n(n+1)/6$ ($\dim(\ker(\sigma)) = 0$, if the r.h.s is negative). This number is: 2, when n = 3, 4; and 0, when n > 5.

Therefore we can not expect $\sum_i \dim((V:X_i)/\mathcal{W}) > (n-1)n(n-1)/2$ when $n \geq 3$, which suggests we have good chance to have $(V:\mathcal{L}) = \mathcal{W}$. Note that this conclusion would be more credible if q or n gets larger.

3.2 Recovering V(g)

In the previous section we have indicated that $f \circ g$ can likely be factored as long as V(g) can be obtained. Now we will show how to get V(g) from the component polynomials, h_1, \dots, h_n , of $f \circ g$.

Let \tilde{V} denote the linear space generated by

$$\frac{\partial h_j}{\partial X_i} \in V(g)$$
, for all i, j .

Lemma 1. $\tilde{V} \subset V(g)$ if q > 3.

Proof. When q > 3, the expression for each h_j as a homogeneous polynomial of degree 4 is unique. We can write h_j in form $\sum a_{k,l}g_kg_l$, so we have

$$\frac{\partial h_j}{\partial X_i} = \sum a_{k,l} \left(\frac{\partial g_k}{\partial X_i} g_l + \frac{\partial g_l}{\partial X_i} g_k \right) \in V(g).$$

Since $\dim(V(g)) \leq n^2$, if we regard the n^2 partial derivatives as random vectors in V(g), then with probability greater than $\prod_{i>0} (1-q^{-i})$, which is close to 1-1/q when q is not too small, we will have $\tilde{V}=V(g)$. In general, the probability that $\dim(V(g)/\tilde{V}) \geq \delta$ is approximately $q^{-\delta^2}$. So when $\tilde{V} \neq V(g)$, we can expect that $\dim(V(g)/\tilde{V})$ be very small, say < n.

When $\tilde{V} \neq V(g)$, V(g) may be recovered from \tilde{V} as follows. Randomly choose a subspace \mathcal{L}' of \mathcal{L} and compute $(\tilde{V}:\mathcal{L}')$, if we can be assured that $(\tilde{V}:\mathcal{L}') \subset \mathcal{L}(g)$, then we can add $(\tilde{V}:\mathcal{L}')\mathcal{L}$ to \tilde{V} , and hope this will enlarge \tilde{V} and by repeating the process to finally get $\tilde{V}=V(g)$. The problem is that it is hard to decide whether $(\tilde{V}:\mathcal{L}') \subset \mathcal{L}(g)$. In the following we will give a solution to this problem for n>4.

Assume $0 < \delta = \dim(V(g)/\tilde{V}) < n$ and n > 4. By arguments in the previous subsection, we have seen that $(V(g):F) = \mathcal{L}(g)$ holds with high probability for a randomly-chosen linear form F. On the other hand, $\dim((F\mathcal{L}(g)) \cap \tilde{V}) \geq n - \delta$, and the equality also holds with high probability. $(V(g):F) = \mathcal{L}(g)$ implies that

$$(\tilde{V}:F) = ((F\mathcal{L}(g)) \cap \tilde{V})/F.$$

So we could expect that $\dim((\tilde{V}:F)) = n - \delta$ occur frequently. Moreover δ can be detected from the fact that

$$\delta = n - \min\{\dim((\tilde{V}:F)) : \text{for sufficiently many random } F\}$$

. Now it is easy to conclude that $(\tilde{V}:F)\subset\mathcal{L}(g)$ for those F satisfying $\dim((\tilde{V}:F))=n-\delta$.

4 An Example

In this section, the methods of the previous section are applied to a concrete example, D^{**} with q=251, n=9, which is suggested in [10]. The irreducible polynomial for definition of $K^{(9)}$ is chosen as t^9+t+8 (the choice is irrelevant to the analysis of the scheme). Let ϕ denote the squaring in $K^{(9)}$. Let $g_s=\phi\circ s$ for any linear bijection s. The property that $(V(g_s):\mathcal{L})=\mathcal{L}(g_s)$ is independent of s. So are the distribution of dimension of $(V(g_s):\mathcal{L}')$ while \mathcal{L}' ranging over subspaces of \mathcal{L} . Therefore in order to verify the properties of $V(g_s)$ as predicted by the heuristic arguments in the previous section, we may assume $g=\phi$. The component polynomials of ϕ is given in appendix, where indexes for variables start with 0. It can be verified that $\dim(V(g))=n^2=81$. We did not find any linear form F, such that $(V(g):F)\neq\mathcal{L}(g)$, among 1000 randomly chosen F. So $\mathcal{L}(g)$ has much stronger properties than that stated in Conjecture 1. This also suggests that, if the inner factor of a "2R" scheme is a one-round scheme of type D^* , the attack described in the previous section would likely be successful.

We have also done experiments to verify that, the linear space V(g) can indeed be recovered by the method described in previous section. For $\pi=t\circ \phi\circ s\circ \phi\circ r$, define \tilde{V}_{π} to be the linear space generated by partial derivatives of component polynomials of π . It is easy to prove that $\dim(\tilde{V}_{\pi})$ does not depend on t and r. So we let t=r=1. Again we have tried 1000 randomly chosen s, and we always get $\dim(\tilde{V}_{\pi})=n^2=81$.

The programs (see Appendix 3) for these experiments are written in Mathematica 3.0, where "test1" tests the properties related to Conjecture 1, and "test2" test the distribution of $\dim(\tilde{V}_{\pi})$.

5 Conclusion

In this paper, we have showed that the functional decomposition problem for compositions of quadratic mappings is not hard provided the field of coefficients has more than 4 elements. As a consequence, the base field for "2R" schemes has only 3 choices: GF(2), GF(3), GF(4). However, in these cases, the dimension n should be large to guarantee a reasonable block size (say, ≥ 64 bits); since the public key size is at the order of n^5 , one can easily see that the resulted schemes are simply impractical. This concludes that the idea of "2R" schemes is not interesting. One possible cure is to replace a few of the component polynomials with random polynomials before composing the last affine bijection, using ideas in [12]. Again, this will greatly reduce the efficiency of private-key side computations, hence lower the practical value of the original designs.

It remains open if the corresponding functional decomposition problem is really hard when $q \leq 4$.

References

- M. Dickerson, The Functional Decomposition of Polynomials, Ph.D Thesis, TR89-1023, Dept. of Computer Science, Cornell University, Ithaca, NY, July 1989.
- W. Diffie and M. E. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory, IT-22(6) 644-654, 1976.
- 3. T.Elgamal, A Public Key Cryptosystem and a Signature Schemes Based on Discrete Logarithms, IEEE Trans. Inform. Theory, Vol. IT-31(1985), 469-472.
- T. Matsumoto and H. Imai, Algebraic Methods for Constructing Asymmetric Cryptosystems, AAECC-3, Grenoble, 1985.
- T. Matsumoto and H. Imai, Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption, Advances in Cryptology, Proceedings of EUROCRYPT'88, Springer Verlag, pp 419-453;
- W.B. Muller, Polynomial Functions in Modern Cryptology, Contributions to General Algebra 3: Proceedings of the Vienna Conference, Vienna: Verlag Holder-Picher-Tempsky, 1985, pp. 7-32.
- J. Patarin, Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypto'88, Advances in Cryptology, Proceedings of CRYPTO'95, Springer Verlag, pp 248-261;
- 8. J. Patarin, Asymmetric Cryptography with a Hidden Monomial, Advances in Cryptology, Proceedings of CRYPTO'96, Springer Verlag, pp 45-60;
- J. Patarin, Hidden Fields Equations and Isomorphisms of Polynomials: Two New Families of Asymmetric Algorithms, Advances in Cryptology, Proceedings of EU-ROCRYPT'96, Springer Verlag, pp 33-48;
- J. Patarin and L. Goubin, Trapdoor one-way permutations and multivariate polynomials, Proceedings of ICICS'97, Lecture Notes in Computer Science, Vol. 1334, Springer, 1997.
- J. Patarin and L.Goubin, Asymmetric cryptography with S-boxes, Proceedings of ICICS'97, Lecture Notes in Computer Science, Vol. 1334, Springer, 1997.
- J. Patarin and L.Goubin, C^{*}₋₊ and HM: Variations Around Two Schemes of T.Matsumoto and H. Imai, Advances in Cryptology, Proceedings of ASI-ACRYPT'98, Lecture Notes in Computer Science 1514, Springer Verlag, pp 35-49.

 R.L. Rivest, A. Shamir, L.M. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of ACM, v.21, n.2, 1978, pp.120-126.

Appendix 1: Homogenlization

For any polynomial $P(X_1, X_2, \dots, X_n)$ of supposed degree $d \geq \deg(P)$, define its homogenization as $\tilde{P} = X_0^d P(X_1/X_0, X_2/X_0, \dots, X_n/X_0)$, where X_0 is a new variable. The supposed degree of the component polynomials of a quadratic mapping is 2, and so on. For any mapping $f: K^n \mapsto K^n$ with component polynomials (f_1, \dots, f_n) , define its homogenization as $\tilde{f} = (X_0^{\deg f}, \tilde{f}_1, \dots, \tilde{f}_n)$. Conversely, for any f of this form, define its dehomogenization to be $f = (\tilde{f}_1(1, X_1, \dots, X_n), \dots, \tilde{f}_n(1, X_1, \dots, X_n))$.

Lemma 2. Let the f, g be two mappings $K^n \to K^n$. If $q > \deg f \times \deg g$, then $f \circ g = \tilde{f} \circ \tilde{g}$.

Proof. In this case, composition of mappings is equivalent to composition of polynomials, and the lemma follows from the fact that homogenization commutes with polynomial composition.

Suppose we are given a "2R" public key π , if we can decompose $\tilde{\pi} = \tilde{f} \circ \tilde{g}$, then the decomposition $\pi = f \circ g$ can be obtained simply by dehomogenization. The above lemma guarantees the existence of such a decomposition of $\tilde{\pi}$. In decomposing $\tilde{\pi}$ using the method of this paper, we should add the n polynomials $X_0^2 X_1, \dots, X_0^2 X_n$ to \tilde{V} , the space of partial derivatives.

Appendix 2: Some Basic Facts of Linear Algebra

1. The number of subspaces of dimension k in a space of dimension n > k is:

$$\mu(k,n) = \prod_{0 \leq i < k} (q^n - q^i) / \prod_{0 \leq i < k} (q^k - q^i) \approx q^{(n-k)k}$$

2. The number of $n \times N$ matrices with rank $\leq k < \min(n, N)$ is less than

$$q^{kN}\mu(k,n) \approx q^{k(n+N-k)}$$

3. The probability that the intersection of two random subspaces of dimension n_1 , n_2 in a space of dimension n has dimension $n_1 + n_2 - n + \delta \ge 0$ ($\delta \ge 0$) is

$$\mu(n - n_2 - \delta, 2n - n_1 - n_2 - \delta)\mu(n - n_1 - \delta, 2n - n_1 - n_2 - \delta)$$

$$\mu(n_1 + n_2 - n + \delta, n)\mu(n_1, n)^{-1}\mu(n_2, n)^{-1} \approx q^{-\delta(n_1 + n_2 - n + \delta)}$$

4. The probability that a random linear mapping $\sigma: K^{n_1} \to K^{n_2}$ has a kernel of dimension $e = \max(n_1 - n_2, 0) + \delta$ $(\delta \ge 0)$ is

$$q^{-n_1 n_2} \mu(e,n_1) \mu(n_1-e,n_2) \prod_{0 \leq i < n_1-e} (q^{n_1-e}-q^i) \approx q^{-e(e+n_2-n_1)}$$

Appendix 3

The following is the source code for our experiments written in Mathematica 3.0.

```
p=251; n=9;
phi={
  x[0]^2 + 235x[4]x[5] + 235x[3]x[6] + 235x[2]x[7] + 235x[1]x[8],
  2x[0]x[1] + 249x[4]x[5] + 243x[5]^2 + 249x[3]x[6] + 235x[4]x[6]
  + 249x[2]x[7] + 235x[3]x[7] + 249x[1]x[8] + 235x[2]x[8],
  x[1]^2 + 2x[0]x[2] + 250x[5]^2 + 249x[4]x[6] + 235x[5]x[6] +
  249x[3]x[7] + 235x[4]x[7] + 249x[2]x[8] + 235x[3]x[8]
  2x[1]x[2] + 2x[0]x[3] + 249x[5]x[6] + 243x[6]^2 + 249x[4]x[7] +
  235x[5]x[7] + 249x[3]x[8] + 235x[4]x[8]
  x[2]^2 + 2x[1]x[3] + 2x[0]x[4] + 250x[6]^2 + 249x[5]x[7] +
  235x[6]x[7] + 249x[4]x[8] + 235x[5]x[8],
  2x[2]x[3] + 2x[1]x[4] + 2x[0]x[5] + 249x[6]x[7] + 243x[7]^2 +
  249x[5]x[8] + 235x[6]x[8],
  x[3]^2 + 2x[2]x[4] + 2x[1]x[5] + 2x[0]x[6] + 250x[7]^2 +
  249x[6]x[8] + 235x[7]x[8],
  2x[3]x[4] + 2x[2]x[5] + 2x[1]x[6] + 2x[0]x[7] + 249x[7]x[8] +
  243x[8]^2, x[4]^2 + 2x[3]x[5] + 2x[2]x[6] + 2x[1]x[7] +
  2x[0]x[8] + 250x[8]^2
  };
tovector2[f_]:=Flatten[Table[Coefficient[f, x[i]x[j]],
                             \{i,0,n-1\},\{j,i,n-1\}];
id=IdentityMatrix[n(n+1)(n+2)/6];
mu[i_,j_,k_]:=Block[{i1,j1,k1},
              If[i<=j, i1=i;j1=j;k1=k, i1=j;</pre>
              If[i<=k, j1=i; k1=k, j1=k; k1=i]];</pre>
              Return[n(n+1)(n+2)/6-(n-i1)(n-i1+1)(n-i1+2)/6
                     +(n-i1)(n-i1+1)/2-(n-i1)(n-i1+1)/2+k1-i1+1];
Do[M[i]=id[[Flatten[Table[mu[i,j,k],{j,0,n-1},{k,j,n-1}]]]],
  {i, 0, n-1}];
H=NullSpace[Table[tovector2[phi[[i]]].M[i], {i,0, n-1}],
           Modulus->p];
Do[check[i]=Transpose[M[i].Transpose[H]], {i,0, n-1}];
rank[L_]:=Length[NullSpace[Sum[L[[i+1]]check[i],{i,0, n-1}],
                           Modulus->p]]-n;
```

```
test1[count_]:=Block[{i,L, r}, i=1;
                 While[i<=count,
                    L=Table[Random[Integer,p-1], {j,n}];
                    r=rank[L];
                    If[r>0, Save["result.mat", {i, r, L}]];
                    i++]];
psi=phi/.Table[x[i]->y[i],{i,0,n-1}];
d[f_, i_]:=Sum[j Coefficient[Collect[f,x[i]],
              x[i]^jx[i]^(j-1), {j,4}];
tovector3[f_]:=Flatten[Flatten[
                  Table [Coefficient [f,x[i]x[j]x[k]],
                        \{i,0,n-1\},\{j,i,n-1\},\{k,j,n-1\}]\};
test2[count_]:=Block[{A, f, n0, n1, n2, i, S,r,h},
                 i=1; n0=n1=n2=0;
                 While[i<=count,
                 A=Table[Random[Integer,p-1], \{k,n\}, \{j,n\}];
                 f=phi.A ; g=Expand[f, Modulus->p]; S={};
                 h=Expand[psi/.Table[y[j]->g[[j+1]],
                         {j,0,n-1}], Modulus->p];
                 Do[AppendTo[S, tovector3[d[h[[k]], j]]],
                        {k,n}, {j,n}];
                 r=n^2-n(n+1)(n+2)/6+Length[NullSpace[S,
                                               Modulus->p]];
                 Switch[r, 0, n0++, 1, n1++, 2, n2++]; i++];
                 Print[n0]; Print[n1]; Print[n2];];
```

Factoring $N = p^r q$ for Large r

Dan Boneh^{1*}, Glenn Durfee^{1**}, and Nick Howgrave-Graham²

Abstract. We present an algorithm for factoring integers of the form $N = p^r q$ for large r. Such integers were previously proposed for various cryptographic applications. When $r \approx \log p$ our algorithm runs in polynomial time (in $\log N$). Hence, we obtain a new class of integers that can be efficiently factored. When $r \approx \sqrt{\log p}$ the algorithm is asymptotically faster than the Elliptic Curve Method. Our results suggest that integers of the form $N = p^r q$ should be used with care. This is especially true when r is large, namely r greater than $\sqrt{\log p}$.

1 Introduction

In recent years moduli of the form $N=p^rq$ have found many applications in cryptography. For example, Fujioke et al. [3] use a modulus $N=p^2q$ in an electronic cash scheme. Okamoto and Uchiyama [12] use $N=p^2q$ for an elegant public key system. Last year Takagi [18] observed that RSA decryption can be performed significantly faster by using a modulus of the form $N=p^rq$. In all of these applications, the factors p and q are approximately the same size. The security of the system relies on the difficulty of factoring N.

We show that moduli of the form $N=p^rq$ should be used with care. In particular, let p and q be primes of a certain length, say 512 bits each. We show that factoring $N=p^rq$ becomes easier as r gets bigger. For example, when r is on the order of $\log p$ our algorithm factors N in polynomial time. This is a new class of moduli that can be factored efficiently. When $N=p^rq$ with r on the order of $\sqrt{\log p}$ our algorithm factors N faster than the current best method—the elliptic curve algorithm (ECM) [10]. Hence, if p and q are 512 bit primes, then $N=p^rq$ with $r\approx 23$ can be factored by our algorithm faster than with ECM. These results suggest that moduli of the form $N=p^rq$ with large r are inappropriate for cryptographic purposes. In particular, Takagi's proposal [18] should not be used with a large r.

Suppose p and q are k bit primes and $N = p^r q$. When $r = k^{\epsilon}$ our algorithm (asymptotically) runs in time $T(k) = 2^{(k^{1-\epsilon})+O(\log k)}$. Hence, when $\epsilon = 1$ the modulus N is roughly k^2 bits long and the algorithm will factor N in polynomial

¹ Computer Science Department, Stanford University, Stanford, CA 94305-9045 {dabo,gdurf}@cs.stanford.edu

² Mathematical Sciences Department, University of Bath, Bath, BA2 7AY, U.K nahg@maths.bath.ac.uk

^{*} Supported in part by NSF CCR-9732754.

^{**} Supported by Certicom.

time in k. Already when $\epsilon = \frac{1}{2}$ the algorithm asymptotically performs better than ECM. The algorithm requires only polynomial space (in log N).

We implemented the algorithm to experiment and compare it to ECM factoring. It is most interesting to compare the algorithms when $\epsilon \approx 1/2$, namely $r \approx \sqrt{\log p}$. Unfortunately, since $N = p^r q$ rapidly becomes to large to handle we could only experiment with small p's. Our largest experiment involves 96 bit primes p and q and r = 9. In this case N is 960 bits long. Our results suggest that although our algorithm is asymptotically superior, for such small prime factors the ECM method is better. Our experimental results are described in Section 4.

The problem of factoring $N = p^r q$ is related to that of factoring moduli of the form $N = p^2 q$. Previous results due to Peralta and Okamato [13] and also due to Pollard and Bleichenbacher show that ECM factoring can be made more efficient when applied to $N = p^2 q$. Our results for $N = p^2 q$ are described in Section 6.

Our approach is based on techniques due to Coppersmith [2]. We use a simplification due to Howgrave-Graham [5,6]. This technique uses results from the theory of integer lattices. The next section provides a brief introduction to lattices. In Section 3, we describe the factoring algorithm for integers of the form $N = p^r q$ for large r. In Section 4 we discuss our implementation of the algorithm and provide examples of factorizations completed. In Section 5, we compare this approach to existing methods, and describe classes of integers for which this algorithm is the best known method.

2 Lattices

Let $u_1, \ldots, u_d \in \mathbb{Z}^n$ be linearly independent vectors with $d \leq n$. A lattice L spanned by $\langle u_1, \ldots, u_d \rangle$ is the set of all integer linear combinations of u_1, \ldots, u_d . We say that the lattice is full rank if d = n. We state a few basic results about lattices and refer to [11] for an introduction.

Let L be a lattice spanned by $\langle u_1, \ldots, u_d \rangle$. We denote by u_1^*, \ldots, u_d^* the vectors obtained by applying the Gram-Schmidt orthogonalization process to the vectors u_1, \ldots, u_d . We define the determinant of the lattice L as

$$\det(L) := \prod_{i=1}^{d} ||u_i^*||.$$

If L is a full rank lattice then the determinant of L is equal to the determinant of the $d \times d$ matrix whose rows are the basis vectors u_1, \ldots, u_d .

Fact 1 (LLL). Let L be a lattice spanned by $\langle u_1, \ldots, u_d \rangle$. Then the LLL algorithm, given $\langle u_1, \ldots, u_d \rangle$, will produce a vector v satisfying

$$||v|| \le 2^{d/2} \det(L)^{1/d}.$$

The algorithm runs in time quartic in the size of its input.

3 Factoring $N = p^r q$

Our goal in this section is to develop an algorithm to factor integers of the form $N = p^r q$. The main theorem of this section is given below. We use non-standard notation and write $\exp(n) = 2^n$. Similarly, throughout the paper all logarithms should be interpreted as logarithms to the base 2.

Theorem 2. Let $N = p^r q$ where $q < p^c$ for some c. The factor p can be recovered from N, r, and c by an algorithm with a running time of:

$$\exp\left(\frac{c+1}{r+c} \cdot \log p\right) \cdot O(\gamma),$$

where γ is the time it takes to run LLL on a lattice of dimension $O(r^2)$ with entries of size $O(r \log N)$. The algorithm is deterministic, and runs in polynomial space.

Note that the factor γ is polynomial in $\log N$. It is worthwhile to consider a few examples using this theorem. For simplicity we assume c=1, so that both p and q are roughly the same size. Taking c as any small constant gives similar results.

- When c=1 we have that $\frac{c+1}{r+c}=O(\frac{1}{r})$. Hence, the larger r is, the easier the factoring problem becomes. When $r=\epsilon\log p$ for a fixed ϵ , the algorithm is polynomial time.
- When $r \approx \log^{1/2} p$, then the running time is approximately $\exp(\log^{1/2} p)$. Thus, the running time is (asymptotically) slightly better than the Elliptic Curve Method (ECM) [10]. A comparison between this algorithm and existing algorithms is given in Section 5.
- For small r, the algorithm runs in exponential time.
- When c is large (e.g. on the order of r) the algorithm becomes exponential time. Hence, the algorithm is most effective when p and q are approximately the same size. All cryptographic applications of $N = p^r q$ we are aware of use p and q of approximately the same size.

The proof of Theorem 2 is based on a technique due to Coppersmith [2] and Howgrave-Graham [5]. The basic idea is to guess a small number of the most significant bits of p and factor using the guess. As it turns out, we can show that the larger r is, the fewer bits of p we need to guess.

In [2] Coppersmith shows that given half the bits of p (the most significant bits) one can factor integers of the form N = pq in polynomial time, provided p and q are about the same size. To do so Coppersmith proved an elegant result showing how to find small solutions of bivariate equations over the integers. Surprisingly, Theorem 2 does not follow from Coppersmith's results. Coppersmith's bivariate theorem does not seem to readily give an efficient algorithm for factoring $N = p^r q$. Recently, Howgrave-Graham [5] showed an alternate way of deriving Coppersmith's results for univariate modular polynomials. He then showed in [6] how the univariate modular results enable one to factor N = pq

given half the most significant bits of p assuming p and q are of the same size. In the case that p and q are of different size, both Coppersmith's and Howgrave-Graham's results are weaker, in the sense of requiring a higher percentage of the bits of the smaller factor to be known.

We prove Theorem 2 by extending the univariate modular approach. Our results are an example in which the modular approach appears to be superior to the bivariate integer approach.

Note that for simplicity we assume r and c are given to the algorithm of Theorem 2. Clearly this is not essential since one can try all possible values for r and c until the correct values are found.

Lattice-based factoring

We are given $N=p^rq$. Suppose that in addition, we are also given an integer P that matches p on a few of p's most significant bits. In other words, |P-p| < X for some large X. For now, our objective is to find p given N, r, and P. Consider the polynomial $f(x)=(P+x)^r$. Then the point $x_0=p-P$ satisfies $f(x_0)\equiv 0$ mod p^r . Hence, we are looking for a root of f(x) modulo p^r satisfying $|x_0| < X$. Unfortunately, the modulus p^r is unknown. Instead, only a multiple of it, N, is known.

Given a polynomial $h(x) = \sum_i a_i x^i$ we define $||h(x)||^2 = \sum_i |a_i^2|$. The main tool we use to find x_0 is stated in the following simple fact which was previously used in [9,4,5].

Fact 3. Let $h(x) \in \mathbb{Z}[x]$ be a polynomial of degree d. Suppose that

a. $h(x_0) \equiv 0 \mod p^{rm}$ for some positive integers r, m where $|x_0| < X$, and b. $||h(xX)|| < p^{rm}/\sqrt{d}$.

Then $h(x_0) = 0$ holds over the integers.

Proof. Observe that

$$|h(x_0)| = \left| \sum a_i x_0^i \right| = \left| \sum a_i X^i \left(\frac{x_0}{X} \right)^i \right| \le$$

$$\sum \left| a_i X^i \left(\frac{x_0}{X} \right)^i \right| \le \sum \left| a_i X^i \right| \le \sqrt{d} \|h(xX)\| < p^{rm},$$

but since $h(x_0) \equiv 0$ modulo p^{rm} we have that $h(x_0) = 0$.

Fact 3 suggests that we should look for a polynomial h(x) that has x_0 as a root modulo p^{rm} , for which h(xX) has norm less than roughly p^{rm} . Let m > 0 be an integer to be determined later. For $k = 0, \ldots, m$ and any $i \ge 0$ define:

$$g_{i,k}(x) := N^{m-k} x^i f^k(x).$$

Observe that x_0 is a root of $g_{i,k}(x)$ modulo p^{rm} for all i and all k = 0, ..., m. We are looking for an integer linear combination of the $g_{i,k}$ of norm less than

 p^{rm} . To do so we form a lattice spanned by the $g_{i,k}(xX)$ and use LLL to find a short vector in this lattice. Once we find a "short enough" vector h(xX) it will follow from Fact 3 that x_0 is a root of h(x) over \mathbb{Z} . Then x_0 can be found using standard root finding methods over the reals.

Let L be the lattice spanned by the coefficients vectors of:

(1)
$$g_{i,k}(xX)$$
 for $k = 0, ..., m-1$ and $i = 0, ..., r-1$, and

(2)
$$g_{i,m}(xX)$$
 for $j = 0, ..., d - mr - 1$.

The values of m and d will be determined later. To use Fact 1, we must bound the determinant of the resulting lattice. Let M be a matrix whose rows are the basis vectors for L (see Figure 1). Notice that M is a triangular matrix, so the determinant of L is just the product of the diagonal entries of M. This is given by

$$\det(M) = \left(\prod_{k=0}^{m-1} \prod_{i=0}^{r-1} N^{m-k}\right) \left(\prod_{j=0}^{d-1} X^j\right) < N^{rm(m+1)/2} X^{d^2/2}.$$

Fact 1 guarantees that the LLL algorithm will find a short vector u in L satisfying

$$||u||^d \le 2^{d^2/2} \det(L) \le 2^{d^2/2} N^{rm(m+1)/2} X^{d^2/2}.$$
 (1)

This vector u is the coefficients vector of some polynomial h(xX) satisfying ||h(xX)|| = ||u||. Furthermore, since h(xX) is an integer linear combination of the polynomials $g_{i,k}(xX)$, we may write h(x) as an integer linear combination of the $g_{i,k}(x)$. Therefore $h(x_0) \equiv 0 \mod p^{rm}$. To apply Fact 3 to h(x) we require that

$$||h(xX)|| < p^{rm}/\sqrt{d}.$$

Example lattice for $N = p^2q$ when m = 3 and d = 9. The entries marked with "*' correspond to non-zero entries whose value we ignore. The determinant of the lattice is the product of the elements on the diagonal. Elements on the diagonal are given explicitly.

Fig. 1. Example of the lattice formed by the vectors $g_{i,k}(xX)$

The factor of \sqrt{d} in the denominator has little effect on the subsequent calculations, so for simplicity it is omitted. Plugging in the bound on ||h(xX)|| from equation (1) and reordering terms, we see this condition is satisfied when:

$$(2X)^{d^2/2} < p^{rmd} N^{-rm(m+1)/2}.$$

Suppose $q < p^c$ for some c. Then $N < p^{r+c}$, so we need

$$(2X)^{d^2/2} < p^{rmd-r(r+c)m(m+1)/2}.$$

Larger values of X allow us to use weaker approximations P, so we wish to find the largest X satisfying the bound. The optimal value of m is attained at $m_0 = \lfloor \frac{d}{r+c} - \frac{1}{2} \rfloor$, and we may choose d_0 so that $\frac{d_0}{r+c} - \frac{1}{2}$ is within $\frac{1}{2r+c}$ of an integer. Plugging in $m = m_0$ and $d = d_0$ and working through tedious arithmetic results in the bound:

$$X < \frac{1}{2}p^{1-\frac{c}{r+c}-\frac{r}{d}(1+\delta)} \qquad \text{where} \qquad \delta = \frac{1}{r+c} - \frac{r+c}{4d}.$$

Since $\delta < 1$ we obtain the slightly weaker, but more appealing bound:

$$X < p^{1 - \frac{c}{r + c} - 2\frac{r}{d}} \tag{2}$$

When X satisfies the bound of equation (2), the LLL algorithm will find in L a vector h(xX) satisfying $||h(xX)|| < p^{rm}/\sqrt{d}$. This short vector will give rise to the polynomial equation h(x), which is an integer linear combination of the $g_{i,k}(x)$ and thus has x_0 as a root modulo p^{rm} . But since ||h(xX)|| is bounded, we have by Fact 3 that $h(x_0) = 0$ over \mathbb{Z} , and normal root-finding methods can extract the desired x_0 . Given x_0 the factor $p = P + x_0$ is revealed.

We summarize this result in the following lemma.

Lemma 1. Let $N = p^r q$ be given, and assume $q < p^c$ for some c. Furthermore assume that P is an integer satisfying:

$$|P - p| < p^{1 - \frac{c}{r + c} - 2\frac{r}{d}}.$$

Then the factor p may be computed from N, r, c, and P by an algorithm whose running time is dominated by the time it takes to run LLL on a lattice of dimension d.

Note that as d tends to infinity the bound on P becomes $|P-p| < p^{1-\frac{c}{r+c}}$. When c=1 taking $d=r^2$ provides a similar bound and is sufficient for practical purposes. We can now complete the proof of the main theorem.

Proof of Theorem 2 Suppose $N = p^r q$ with $q < p^c$ for some c. Let d = 2r(r+c). Then, by Lemma 1 we know that given an integer P satisfying

$$|P - p| < p^{1 - \frac{c+1}{r+c}}$$

the factorization of N can be found in time $O((\log N)^2 d^4)$. Let $\epsilon = \frac{c+1}{r+c}$. We proceed as follows:

- a. For all $k = 1, \ldots, (\log N)/r$ do:
- b. For all $j = 0, \ldots, 2^{\epsilon k}$ do:
- c. Set $P = 2^k + i \cdot 2^{(1-\epsilon)k}$.
- d. Run the algorithm of Lemma 1 using the approximation P.

The outer most loop on the length of p is not necessary if the size of p is known. If p is k bits long then one of the P's generated in step (c) will satisfy $|P-p| < 2^{(1-\epsilon)k}$ and hence $|P-p| < p^{1-\epsilon}$ as required. Hence, the algorithm will factor N in the required time.

4 Implementation and Experiments

We implemented the lattice factoring method (LFM) using Maple version 5.0 and Victor Shoup's NTL (Number Theory Library) package [17]. The program operates in two phases. First, it guesses the most significant bits P of the factor p, then builds the lattice described in Section 3. Using NTL's implementation of LLL, it reduces the lattice from Section 3, looking for short vectors. Second, once a short vector is found, the corresponding polynomial is passed to Maple, which computes the roots for comparison to the factorization of N.

Several observations were made in the implementation of this algorithm. First of all, it was found that the order in which the basis vectors appear in the lattice given to LLL matters. In particular, since the final polynomial is almost always of degree equal to the dimension of the lattice, this means that a linear combination which yields a short vector must include those basis vectors corresponding to the last few $g_{i,k}$, say $k = m/2, \ldots, m$ and $i = 0, \ldots, r$. It turns out to be beneficial to place them at the "top" of the lattice, where LLL would perform row reduction first, as these alone would likely be enough to produce a short vector. We found the optimal ordering for the $g_{i,k}$ to be $i = r - 1, \ldots, 0$, $k = m, m - 1, \ldots, 0$; this resulted in greatly reduced running time compared to the natural ordering, in which LLL spent a large amount of time reducing basis vectors that would ultimately be irrelevant.

The reader may have noticed that in building the lattice in Section 3, we could have taken powers of (P+x) instead of shifts and powers of $(P+x)^r$. The reason for performing the latter is mainly for a performance improvement. Although both methods yield a lattice with the same determinant, using shifts and powers of $(P+x)^r$ produces a matrix that appears "more orthogonal". That is, certain submatrices of the matrix from Section 3 are Toeplitz matrices, and heuristically this should make it easier for LLL to find a good basis. We compared both methods and found a speedup of about ten percent by working with $(P+x)^r$.

Lastly, recall that in an LLL-reduced lattice, the shortest vector u satisfies

$$||u|| < 2^{d/2} \det(L)^{1/d}$$
.

Implementations of LLL often try to improve on this "fudge factor" of $2^{d/2}$. However, as the analysis from Section 3 shows, its effect is negligible, requiring

only an extra bit of p to be known. Therefore, the higher-quality reduction produced with a smaller fudge factor is not necessary, and running times can be greatly improved by "turning off" improvements such as block Korkin-Zolotarev reduction.

To test the algorithm, we assumed that an approximation P of the desired quality was given; we model this by "giving" the appropriate number of bits to the algorithm before constructing the lattice. In general, these bits would be exhaustively searched, so k bits given would imply a running time of 2^k times what is shown. We ran several experiments and have listed the results in Figure 2. Needless to say, the results by themselves are not so impressive; for such small N, ECM performs much better. However, we expect the running time to scale polynomially with the size of the input, quickly outpacing the running times of ECM and NFS, which scale much less favorably.

p	N	r	bits given	lattice dimension	running time
64 bits	576 bits	8	16 bits	49	20 minutes
80 bits	1280 bits	15	20 bits	72	21 hours
96 bits	768 bits	7	22 bits	60	7 hours
96 bits	960 bits	9	22 bits	65	10 hours
100 bits	600 bits	5	23 bits	69	11 hours

Fig. 2. Example running times on a 400MhZ Pentium running Windows NT

5 Comparison to Other Factoring Methods

We restate Theorem 2 so that it is easier to compare lattice factoring to existing algorithms. We first introduce some notation. Let $T_{\alpha}(p)$ be the function defined by:

$$T_{\alpha}(p) = \exp\left((\log p)^{\alpha}\right)$$

This function is analogous to the $L_{\alpha,\beta}(p)$ function commonly used to describe the running time of factoring algorithms [8]. Recall that

$$L_{\alpha,\beta}(p) = \exp\left(\beta(\log p)^{\alpha}(\log\log p)^{1-\alpha}\right)$$

One can easily see that $T_{\alpha}(p)$ is slightly smaller than $L_{\alpha,1}(p)$. We can now state a special case of Theorem 2.

Corollary 1. Let $N = p^r q$ be given where p and q are both k bit integers. Suppose $r = (\log p)^{\epsilon}$ for some ϵ . Then given N and r, a non-trivial integer factor of N can be found in time

$$\gamma \cdot T_{1-\epsilon}(p) = \exp\left[(\log p)^{1-\epsilon}\right] \cdot \gamma$$

where γ is polynomial in $\log N$.

Asymptotic Comparison

Let p, q be k-bit primes, and suppose we are given $N = p^r q$. We study the running time of various algorithms with respect to k and r, and analyze their behaviors as r goes to infinity. We write $r = (\log p)^{\epsilon}$. The standard running times [1,7] of several algorithms are summarized in the following table, ignoring polynomial factors.

Method	Asymptotic running time	
Lattice Factoring Method	$\exp\left((\log p)^{1-\epsilon}\right)$	
Elliptic Curve Method	$\exp\left(1.414 \cdot (\log p)^{1/2} (\log \log p)^{1/2}\right)$	
Number Field Sieve	$\exp\left(1.902 \cdot (\log N)^{1/3} (\log \log N)^{2/3}\right)$	

Since $N = p^r q$ and $r = k^{\epsilon}$, we know that

$$\log N = r \log p + \log q \ge rk = k^{1+\epsilon}.$$

Rewriting the above running times in terms of k yields the following list of asymptotic running times.

Method	Asymptotic running time
Lattice Factoring Method	$\exp\left(k^{1-\epsilon}\right) = T_{1-\epsilon}(p)$
Elliptic Curve Method	$\exp\left(1.414 \cdot k^{1/2} (\log k)^{1/2}\right) > (T_{1/2}(p))^{1.414}$
Number Field Sieve	$\exp\left(1.902 \cdot k^{(1+\epsilon)/3} ((1+\epsilon)\log k)^{2/3}\right)$
	$> (T_{(1+\epsilon)/3}(p))^{1.902}$

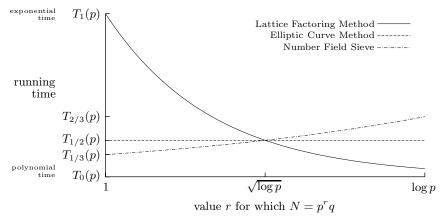
We are particularly interested in the exponential component of the running times, which is tracked in Figure 3. Notice that when $\epsilon = \frac{1}{2}$, then all three algorithms run in time close to $T_{1/2}(p)$.

Practical Comparison to ECM

Of particular interest in Figure 3 is the point at $r = \sqrt{\log p}$ (i.e. $\epsilon = \frac{1}{2}$), where ECM, LFM, and NFS have similar asymptotic running times. We refer the reader to Figure 2 for the sample running times with the lattice factoring method on similar inputs.

Since some of the larger integers that we are attempting to factor exceed 1000 bits, it is unlikely that current implementations of the Number Field Sieve will perform efficiently. This leaves only the Elliptic Curve Method for a practical comparison. Below, we reproduce a table of some example running times [19,15] for factorizations performed by ECM.

size of p	running time with $r=1$	predicted run time for large r
64 bits	53 seconds	r = 8: 848 seconds
96 bits	2 hours	r = 9: 50 hours
128 bits	231 hours	r = 10: 7000 hours



Comparison of subexponential running times of current factoring methods as a function of r. Both axes are logarithmic, and polynomial time factors are suppressed.

Fig. 3. Asymptotic comparison the lattice factoring method with ECM and NFS

Clearly, the Elliptic Curve Method easily beats the lattice factoring method for small integers. However, LFM scales polynomially while ECM scales exponentially. Based on the two tables above we conjecture that the point at which our method will be faster than ECM in practice is for $N = p^r q$ where p and q are somewhere around 400 bits and $r \approx 20$.

6 An Application to Integers of the Form $N = p^2q$

Throughout this section we assume that p and q are two primes of the same size. Lemma 1 can be used to obtain results on "factoring with a hint" for integers of the form $N = p^r q$ for small r. Coppersmith's results [2] show that when N = pq a hint containing half the bits of p is sufficient to factor N. When r = 1, Lemma 1 reduces to the same result. However, when r = 2, i.e. $N = p^2 q$, the lemma shows that only a third of the bits of p are required to factor N. In other words, the hint need only be of the size of $N^{1/9}$. Hence, moduli of the form $N = p^2 q$ are more susceptible to attacks (or designs) that leak bits of p.

7 Conclusions

We showed that for cryptographic applications, integers of the form $N=p^rq$ should be used with care. In particular, we showed that the problem of factoring such N becomes easier as r get bigger. For example, when $r=\epsilon\log p$ for a fixed constant $\epsilon>0$ the modulus N can be factored in polynomial time. Hence, if p and q are k bit primes, the modulus $N=p^kq$ can be factored by a polynomial time algorithm. Even when $r\approx \sqrt{\log p}$ such moduli can be factored in time that

is asymptotically faster than the best current methods. Our results say very little about the case of small r, e.g. when $N = p^2 q$.

Our experiments show that when the factors p and q are small (e.g. under 100 bits) the algorithm is impractical and cannot compete with the ECM. However, the algorithm scales better; we conjecture that as soon as p and q exceed 400 bits each, it performs better than ECM when r is sufficiently large.

Surprisingly, our results do not seem to follow directly from Coppersmith's results on finding small roots of bivariate polynomials over the integers. Instead, we extend an alternate technique due to Howgrave-Graham. It is instructive to compare our results to the case of unbalanced RSA where N=pq is the product of two primes of different size, say p is much larger than q. Suppose p is a prime on the order of q^s . Then, the larger s is, the *more* bits of q are needed to efficiently factor N. In contrast, we showed that when $N=p^rq$, the larger r is, the *fewer* bits of p are needed.

One drawback of the lattice factoring method is that for each guess of the most significant bits of p, the LLL algorithm has to be used to reduce the resulting lattice. It is an interesting open problem to devise a method that will enable us to run LLL once and test multiple guesses for the MSBs of p. This will significantly improve the algorithm's running time. A solution will be analogous to techniques that enable one to try multiple elliptic curves at once in the ECM. Another question is to generalize the LFM to integers of the form $N = p^r q^s$ where r and s are approximately the same size.

Acknowledgments

We thank Paul Zimmermann, Peter Montgomery and Bob Silverman for supplying the running times for ECM, and Don Coppersmith for several useful conversations.

References

- D. Coppersmith, "Modifications to the number field sieve", J. of Cryptology, Vol. 6, pp. 169–180, 1993.
- 2. D. Coppersmith, "Small solutions to polynomial equations, and low exponent RSA vulnerabilities", J. of Cryptology, Vol. 10, pp. 233–260, 1997.
- 3. A. Fujioke, T. Okamoto, S. Miyaguchi, "ESIGN: an efficient digital signature implementation for smartcards", In. proc. Eurocrypt '91, pp. 446–457, 1991.
- 4. J. Hastad, "Solving simultaneous modular equations of low degree", SIAM J. of Computing, Vol. 17, No. 2, pp. 336–341, 1988.
- N. Howgrave-Graham, "Finding small roots of univariate modular equations revisited", Proc. of Cryptography and Coding, LNCS 1355, Springer-Verlag, 1997, pp. 131–142.
- N. Howgrave-Graham, "Extending LLL to Gaussian integers", Unpublished Manuscript, March 1998. http://www.bath.ac.uk/~mapnahg/pub/gauss.ps
- A. Lenstra, H.W. Lenstra Jr., "Algorithms in Number Theory", in Handbook of Theoretical Computer Science (Volume A: Algorithms and Complexity), ch. 12, pp. 673–715, 1990.

- 8. A. Lenstra, H.W. Lenstra Jr., "The development of the number field sieve", Lecture Notes in Mathematics, Vol. 1554, Springer-Verlag, 1994.
- A. Lenstra, H.W. Lenstra Jr., and L. Lovasz, "Factoring polynomial with rational coefficients", Mathematiche Annalen, 261:515–534, 1982.
- H.W. Lenstra Jr., "Factoring integers with elliptic curves", Annuals of Mathematics, 126:649-673, 1987.
- L. Lovasz, "An algorithmic theory of numbers, graphs and convexity", SIAM lecture series, Vol. 50, 1986.
- T. Okamoto, S. Uchiyama, "A new public key cryptosystem as secure as factoring", in Proc. Eurocrypt '98, pp. 310–318, 1998.
- R. Peralta, T. Okamoto, "Faster factoring of integers of special form", IEICE Trans. Fundamentals, Vol. E79-A, No. 4, pp. 489-493, 1996.
- J.J. Quisquater and C. Couvreur, "Fast deciperhment algorithm for RSA public-key cryptosystem", *Electronic Letters*, 18(21), pp. 905–907, 1982.
- R. Silverman, Wagstaff Jr., "A Practical analysis of the elliptic curve factoring algorithm", Math. Comp. Vol 61, 1993.
- A. Shamir, "RSA for Paranoids", RSA Laboratories' CryptoBytes, vol. 1, no. 3, pp. 1–4, 1995.
- V. Shoup, Number Theory Library (NTL), http://www.cs.wisc.edu/~shoup/ntl.
- 18. T. Takagi, "Fast RSA-type cryptosystem modulo p^kq ", in Proc. Crypto '98, pp. 318–326, 1998.
- 19. P. Zimmerman, private communications.

An Efficient Public Key Traitor Tracing Scheme

(Extended Abstract)

Dan Boneh Matthew Franklin dabo@cs.stanford.edu franklin@parc.xerox.com

Abstract. We construct a public key encryption scheme in which there is one public encryption key, but many private decryption keys. If some digital content (e.g., a music clip) is encrypted using the public key and distributed through a broadcast channel, then each legitimate user can decrypt using its own private key. Furthermore, if a coalition of users collude to create a new decryption key then there is an efficient algorithm to trace the new key to its creators. Hence, our system provides a simple and efficient solution to the "traitor tracing problem". Our tracing algorithm is deterministic, and catches all active traitors while never accusing innocent users, although it is only partially "black box". A minor modification to the scheme enables it to resist an adaptive chosen ciphertext attack. Our techniques apply error correcting codes to the discrete log representation problem.

1 Introduction

Consider the distribution of digital content to subscribers over a broadcast channel. Typically, the distributor gives each authorized subscriber a hardware or software decoder ("box") containing a secret decryption key. The distributor then broadcasts an encrypted version of the digital content. Authorized subscribers are able to decrypt and make use of the content. This scenario comes up in the context of pay-per-view television, and more commonly in web based electronic commerce (e.g. broadcast of online stock quotes or broadcast of proprietary market analysis).

However, nothing prevents a legitimate subscriber from giving a copy of her decryption software to someone else. Worse, she might try to expose the secret key buried in her decryption box and make copies of the key freely available. The "traitor" would thus make all of the distributor's broadcasts freely available to non-subscribers. Chor, Fiat and Naor [4] introduced the concept of a traitor tracing scheme to discourage subscribers from giving away their keys. Their approach is to give each subscriber a distinct set of keys that both identify the subscriber and enable her to decrypt. In a sense, each set of keys is a "watermark" that traces back to the owner of a particular decryption box. A coalition of traitors might try to mix keys from many boxes, to create a new pirate box that can still decrypt but cannot be traced back to them. A traitor tracing scheme is "k-collusion resistant" if at least one traitor can always be identified when k of them try to cheat in this way. In practice, especially with tamper-resistant

decryption boxes, it may suffice for k to be a fairly small integer, e.g., on the order of 20.

In this paper we present an efficient public key traitor tracing scheme. The public key settings enable anyone to broadcast encrypted information to the group of legitimate receivers. Previous solutions were combinatorial with probabilistic tracing [4, 10, 14–16], and could be either public-key or symmetric-key. Our approach is algebraic with deterministic tracing, and is inherently public-key. Our approach is much more efficient than the public-key instantiations of previous combinatorial constructions.

Previous approaches [4,10] incur an overhead that is proportional to the logarithm of the size of the population of honest users. In a commercial setting such as a web broadcast or pay-per-view tv, where the number of subscribers might be in the millions, this is a significant factor. Our approach eliminates this factor. Furthermore, secret keys in our scheme are very short. Each private key is just the discrete log of a single element of a finite field (e.g., as small as 160 bits in practice). The size of an encrypted message is just 2k + 1 elements of the finite field. The work required to encrypt is about 2k + 1 exponentiations. Decryption takes far less than 2k + 1 exponentiations. During decryption, only the final exponentiation uses the private key, which can be helpful when the secret is stored on a weak computational device.

Previous probabilistic tracing methods try to maximize the chance of catching just *one* of the traitors while minimizing the chance of accusing an innocent user. Our tracing method is deterministic. It catches *all* of the traitors who contributed to the attack. Innocent users are never accused, as long as the number of colluders is at or below the collusion threshold. Even when more than k (but less than 2k) traitors collude, some information about the traitors can be recovered. However, unlike previous tracing methods, our approach is only partially "black box". This limits the tracer in certain scenarios where the pirate decoder can be queried, but the pirate keys cannot be extracted (see Section 4.2).

The intuition behind our system is as follows. Each private key is a different solution vector for the discrete log representation problem with respect to a fixed base of field elements. We can show that the pirate is limited to forming new keys by taking convex combinations of stolen keys. If every set of 2k keys is linearly independent, then every convex combination of k keys can be traced uniquely (but not necessarily efficiently). By deriving our keys from a Reed-Solomon code in the appropriate way, we can take advantage of efficient error correction methods to trace uniquely and efficiently. We note that the multi-dimensional discrete log representation problem has been previously used, e.g., for incremental hashing [1] and Signets [6].

Our scheme is traceable if the discrete log problem is hard. The encryption scheme is secure (semantic security against a passive adversary) if the decision Diffie-Hellman problem is hard. A small modification yields security against an adaptive chosen ciphertext attack under the same hardness assumption. That level of protection can be important in distribution scenarios where the decryption boxes (or decryption software) are widely deployed and largely unsupervised.

In Section 2 we give definitions for the traitor tracing problem. Our basic scheme is described in Section 3. The tracing algorithm is detailed in Section 4. Chosen ciphertext security is considered in Section 5. Conclusions are given in Section 7, including an application of our scheme to defending against software piracy.

2 Definitions

For a detailed presentation of the traitor tracing model, see [4]. A public key traitor tracing encryption scheme is a public key encryption system in which there is a unique encryption key and multiple decryption keys. The scheme is made up of four components:

Key Generation: The key generation algorithm takes as input a security parameter s and a number ℓ of private keys to generate. It outputs a public encryption key e and a list of private decryption keys d_1, \ldots, d_{ℓ} . Any decryption key can be used to decrypt a ciphertext created using the encryption key.

Encryption: The encryption algorithm takes a public encryption key e and a message M and outputs a ciphertext C.

Decryption: The decryption algorithm takes a ciphertext C and any of the decryption keys d_i and outputs the message M. This is an "open" scheme in the sense that only the short decryption keys are secret while the decryption method can be public.

Tracing: Suppose a pirate gets hold of k decryption keys d_1, \ldots, d_k . Using the k keys he creates a pirate decryption box (or decryption software) \mathcal{D} . The encryption scheme is said to be "k-resilient" if there is a tracing algorithm that can determine at least one of the d_i 's in the pirate's possession. The tracing algorithm is said to be "black box" if its only use of \mathcal{D} is as an oracle to query on various inputs.

Representations: Our traitor tracing scheme relies on the representation problem. When $y = \prod_{i=1}^{2k} h_i^{\delta_i}$ we say that $(\delta_1, \dots, \delta_{2k})$ is a "representation" of y with respect to the base h_1, \dots, h_{2k} . If $\bar{d}_1, \dots, \bar{d}_m$ are representations of y with respect to the same base, then so is any "convex combination" of the representations: $\bar{d} = \sum_{i=1}^m \alpha_i \bar{d}_i$ where $\alpha_1, \dots, \alpha_m$ are scalars such that $\sum_{i=1}^m \alpha_i = 1$.

3 The encryption scheme

We are now ready to present our tracing traitor encryption scheme. Let s be a security parameter and k be the maximal coalition size. Our scheme defends against any collusion of at most k parties. We wish to generate one public key and ℓ corresponding private keys. Without loss of generality we assume $\ell \geq 2k+2$ (if $\ell < 2k+2$ we set $\ell = 2k+2$ and generate ℓ private keys).

Our scheme makes use of a certain linear space tracing code Γ which is a collection of ℓ codewords in \mathbb{Z}^{2k} . The construction of the set Γ and the properties it has to satisfy are described in the next section. For now, it suffices to view the ℓ words in Γ as vectors of integers of length 2k. The set $\Gamma = \{\gamma^{(1)}, \ldots, \gamma^{(\ell)}\}$ is fixed and publicly known.

Let G_q be a group of prime order q. The security of our encryption scheme relies on the difficulty of computing discrete log in G_q . More precisely, the security is based on the difficulty of the Decision Diffie-Hellman problem [3] in G_q as discussed below. One can take as G_q the subgroup of \mathbb{Z}_p^* of order q where p is a prime with q|p-1. Alternatively, one can use the group of points of an elliptic curve over a finite field.

Key generation: Perform the following steps:

- 1. Let $g \in G_q$ be a generator of G_q .
- 2. For $i = 1, \ldots, 2k$ choose a random $r_i \in \mathbb{Z}_q$ and compute $h_i = g^{r_i}$.
- 3. The public key is $\langle y, h_1, \ldots, h_{2k} \rangle$, where $y = \prod_{i=1}^{2k} h_i^{\alpha_i}$ for random $\alpha_1, \ldots, \alpha_{2k} \in \mathbb{Z}_q$.
- 4. A private key is an element $\theta_i \in \mathbb{Z}_q$ such that $\theta_i \cdot \gamma^{(i)}$ is a representation of y with respect to the base h_1, \ldots, h_{2k} . The i'th key, θ_i , is derived from the i'th codeword $\gamma^{(i)} = (\gamma_1, \ldots, \gamma_{2k}) \in \Gamma$ by

$$\theta_i = (\sum_{j=1}^{2k} r_j \alpha_j) / (\sum_{j=1}^{2k} r_j \gamma_j) \pmod{q}$$
(1)

To simplify the exposition we frequently refer to the private key as being the representation $\bar{d}_i = \theta_i \cdot \gamma^{(i)}$. Note however that only θ_i needs to be kept secret since the code Γ is public. One can verify that \bar{d}_i is indeed a representation of y with respect to the base h_1, \ldots, h_{2k} .

Encryption: To encrypt a message M in G_q do the following: first pick a random element $a \in \mathbb{Z}_q$. Set the ciphertext C to be

$$C = \langle M \cdot y^a, h_1^a, \dots, h_{2k}^a \rangle$$

Decryption: To decrypt a ciphertext $C = \langle S, H_1, \dots, H_{2k} \rangle$ using user i'th secret key, θ_i , compute

$$M = S/U^{\theta_i}$$
 where $U = \prod_{j=1}^{2k} H_j^{\gamma_j}$

Here $\gamma^{(i)} = (\gamma_1, \dots, \gamma_{2k}) \in \Gamma$ is the codeword from which θ_i is derived. The cost of computing U is far less than 2k + 1 exponentiations thanks to simultaneous

¹ A codeword might not have an associated private key in the extremely unlikely event that the denominator is zero in the calculation of θ_i .

multiple exponentiation [9, p. 618]. Also note that U can be computed without knowledge of the private key, leaving only a single exponentiation by the private key holder to complete the decryption.

Before going any further we briefly show that the encryption scheme is sound, i.e. any private key θ_i correctly decrypts any ciphertext. Given a ciphertext $C = \langle M \cdot y^a, h_1^a, \ldots, h_{2k}^a \rangle$, decryption will yield $M \cdot y^a / U^{\theta_i}$ where $U = \prod_{j=1}^{2k} (h_j^a)^{\gamma_j}$. Then

$$U^{\theta_i} = (\prod_{j=1}^{2k} g^{a r_j \gamma_j})^{\theta_i} = (g^{\sum_{j=1}^{2k} r_j \gamma_j})^{\theta_i a} = (g^{\sum_{j=1}^{2k} r_j \alpha_j})^a = (\prod_{j=1}^{2k} h_j^{\alpha_j})^a = y^a$$

as needed. The third equality follows from Equation (1). More generally, it is possible to decrypt given any representation $(\delta_1, \ldots, \delta_{2k})$ of y with respect to the base h_1, \ldots, h_{2k} , since $\prod_{j=1}^{2k} (h_j^a)^{\delta_j} = y^a$.

Tracing algorithm: We describe our tracing algorithm in Section 4.

3.1 Proof of security

We now show that our encryption scheme is semantically secure against a passive adversary assuming the difficulty of the Decision Diffie-Hellman problem (DDH) in G_q . The assumption says that in G_q , no polynomial time statistical test can distinguish with non negligible advantage between the two distributions $D = \langle g_1, g_2, g_1^a, g_2^a \rangle$ and $R = \langle g_1, g_2, g_1^a, g_2^b \rangle$ where g_1, g_2 are chosen at random in G_q and $g_1, g_2, g_2, g_3, g_4, g_5$ are chosen at random in g_q and g_q are chosen at random in g_q .

Theorem 1. The encryption scheme is semantically secure against a passive adversary assuming the difficulty of DDH in G_q .

Proof. Suppose the scheme is not semantically secure against a passive adversary. Then there exists an adversary that given the public key $\langle y, h_1, \ldots, h_{2k} \rangle$ produces two messages $M_0, M_1 \in G_q$. Given the encryption C of one of these messages the adversary can tell with non-negligible advantage ϵ which of the two messages he was given. We show that such an adversary can be used to decide DDH in G_q . Given $\langle g_1, g_2, u_1, u_2 \rangle$ we perform the following steps to determine if it is chosen from R or D:

Step 1: Choose random $r_2, \ldots, r_{2k} \in \mathbb{Z}_q$. Set $y = g_1, h_1 = g_2$, and $h_i = g_2^{r_i}$ for $i = 2, \ldots, 2k$.

Step 2: Give $\langle y, h_1, \ldots, h_{2k} \rangle$ to the adversary. Adversary returns $M_0, M_1 \in G_q$. **Step 3:** Pick a random $b \in \{0,1\}$ and construct the ciphertext

$$C = \langle M_b u_1, u_2, u_2^{r_2}, \dots, u_2^{r_{2k}} \rangle$$

Step 4: Give the ciphertext C to the adversary. Adversary returns $b' \in \{0, 1\}$. **Step 5:** If b = b' output "D". Otherwise output "R".

Observe that if the tuple $\langle g_1, g_2, u_1, u_2 \rangle$ is chosen from D, then the ciphertext C is an encryption of M_b . If the quadruple is from R, then the ciphertext is an encryption of $M_b g_1^{(a_1-a_2)}$, where $u_1 = g_1^{a_1}$ and $u_2 = g_2^{a_2}$. In other words, the ciphertext is the encryption of a random message. Hence b = b' holds with probability 1/2. By a standard argument, a non-negligible success probability for the adversary implies a non-negligible success probability in deciding DDH. \square

3.2 Constructing new representations

To decrypt, it suffices to know any representation of y with respect to the base h_1, \ldots, h_{2k} . We have already noted that if $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{Z}_q^{2k}$ are representations of y then any convex combination of $\bar{d}_1, \ldots, \bar{d}_m$ is also a representation of y. The following lemma shows that convex combinations are the only new representations of y that can be efficiently constructed from $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{Z}_q^{2k}$.

Lemma 1. Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be a public key. Suppose an adversary is given the public key and m private keys $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{Z}_q^{2k}$ for m < 2k. If the adversary can generate a new representation \bar{d} of y with respect to the base h_1, \ldots, h_{2k} that is not a convex combination of $\bar{d}_1, \ldots, \bar{d}_m$ then the adversary can compute discrete logs in G_q .

Proof. Let g be a generator of G_q . Suppose we are given $z=g^x$. We show how to use the adversary to compute x. Choose random $a,b,r_1,\ldots,r_m,s_1,\ldots,s_{2k}\in\mathbb{Z}_q$. Construct the set $\{h_1,\ldots,h_{2k}\}$ where $h_i=z^{r_i}g^{s_i}$ for $1\leq i\leq m$ and $h_i=g^{s_i}$ for $m+1\leq i\leq 2k$. Compute $y=z^ag^b$. Find m linearly independent (and otherwise random) solutions $\bar{\alpha}_1,\ldots,\bar{\alpha}_m$ to $\bar{\alpha}\cdot\bar{r}=a$ mod q. Extend these to be (otherwise random) solutions to $\bar{\alpha}\cdot\bar{s}=b$ mod q, while keeping the first m entries of each $\bar{\alpha}_i$ unchanged. These m extended vectors are representations of g with respect to the base g0, g1, g2, g3. Suppose that the adversary can find another representation g3 that is not a convex combination of g4, g5, g7, g8, g8, g8, g9, g9,

4 Linear space tracing

4.1 The tracing algorithm

We now turn our attention to the tracing algorithm for the encryption scheme of Section 3. Throughout this subsection we assume the pirate decoder contains at least one representation of y. Furthermore, we assume that by examining the decoder implementation it is possible to obtain one of these representations, \bar{d} . In Section 4.2 we show that, in some cases, these assumptions are unnecessary.

Suppose the pirate obtains k keys $\bar{d}_1, \ldots, \bar{d}_k$. By Lemma 1, \bar{d} found in the pirate decoder must lie in the linear span of the representations $\bar{d}_1, \ldots, \bar{d}_k$. We construct a tracing algorithm that given \bar{d} outputs one of $\bar{d}_1, \ldots, \bar{d}_k$.

Recall that the construction of private keys made use of a set $\Gamma \subseteq \mathbb{Z}_q^{2k}$ containing ℓ codewords. Each of the ℓ users is given a private key $\bar{d}_i \in \mathbb{Z}_q^{2k}$ which is a multiple of a codeword in Γ . To solve the tracing problem we must construct a set $\Gamma \subseteq \mathbb{Z}_q^{2k}$ containing ℓ codewords with the following property. Let \bar{d} be a point in the linear span of some k codewords $\gamma^{(1)}, \ldots, \gamma^{(k)} \in \Gamma$. Then at least one γ in $\gamma^{(1)}, \ldots, \gamma^{(k)}$ must be a member of any coalition (of at most k users) that can create \bar{d} . This γ identifies one of the private keys that must have participated in the construction of the pirated key \bar{d} . Furthermore, there should exist an efficient tracing algorithm that when given \bar{d} as input, outputs γ . In fact, our tracing algorithm will output every $\gamma^{(i)}$ that has nonzero weight in the linear combination.

The set Γ : We begin by describing the set Γ containing ℓ codewords over \mathbb{Z}_q^{2k} . Since q is a large prime we may assume $q > \max(\ell, 2k)$. Consider the following $(\ell - 2k) \times \ell$ matrix:

$$A = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & \ell \\ 1^2 & 2^2 & 3^2 & \dots & \ell^2 \\ 1^3 & 2^3 & 3^3 & \dots & \ell^3 \\ & \vdots & & & \vdots \\ 1^{\ell-2k-1} & 2^{\ell-2k-1} & 3^{\ell-2k-1} & \dots & \ell^{\ell-2k-1} \end{pmatrix} \pmod{q}$$

Observe that any vector in the span of the rows of A corresponds to a polynomial of degree at most $\ell - 2k - 1$ evaluated at the points $1, \ldots, \ell$.

Let b_1, \ldots, b_{2k} be a basis of the linear space of vectors satisfying $A\bar{x} = 0 \mod q$. Viewing these 2k vectors as the columns of a matrix we obtain an $\ell \times 2k$ matrix B:

$$B = \begin{pmatrix} | & | & | & | \\ b_1 & b_2 & b_3 \dots b_{2k} \\ | & | & | & | \end{pmatrix}$$

We define Γ as the set of rows of the matrix B. Hence, Γ contains ℓ codewords each of length 2k. We note that using Lagrange interpolation one can directly construct the i'th codeword in Γ using approximately ℓ arithmetic operations modulo q.

The tracing algorithm: Consider the set of vectors in Γ . Let $\bar{d} \in \mathbb{Z}_q^{2k}$ be a vector formed by taking a linear combination of at most k vectors in Γ . We show that given \bar{d} one can efficiently determine the unique set of vectors in Γ used to construct \bar{d} . Since the vectors in Γ form the rows of the matrix B above we know there exists a vector $\bar{w} \in \mathbb{F}_q^{\ell}$ of Hamming weight at most k such that $\bar{w} \cdot B = \bar{d}$. We show how to recover the vector \bar{w} given \bar{d} .

Step 1: Find a vector $\bar{v} \in \mathbb{F}_q^{\ell}$ such that $\bar{v} \cdot B = \bar{d}$. Many such vectors exist. Choose one arbitrarily. Since $(\bar{v} - \bar{w}) \cdot B = 0$ we know that $\bar{v} - \bar{w}$ is in

If B is in canonical form with the identity matrix as its first 2k rows, then $\bar{v} = (\bar{d}||0...0)$ suffices.

the linear span of the rows of the matrix A (the rows of A span the space of vectors orthogonal to the columns of B). In other words, there exists a unique polynomial $f \in \mathbb{F}_q[x]$ of degree at most $\ell - 2k - 1$ such that $\bar{v} - \bar{w} = \langle f(1), \ldots, f(\ell) \rangle$.

Step 2: Since \bar{w} has Hamming weight at most k, we know that $\langle f(1), \ldots, f(\ell) \rangle$ equals \bar{v} in all but k components. Hence, using Berlekamp's algorithm [2] we can find f from \bar{v} . The polynomial f gives us the vector $\bar{v} - \bar{w}$ from which we recover \bar{w} as required.

For completeness we briefly recall Berlekamp's algorithm. The algorithm enables us to find f given the vector $\bar{v} \in \mathbb{Z}_q^\ell$. Let g be a polynomial of degree at most k such that g(i) = 0 for all $i = 1, \ldots, \ell$ for which $f(i) \neq v_i$ (where v_i is the i'th component of \bar{v}). Then we know that for all $i = 1, \ldots, \ell$ we have $f(i)g(i) = g(i)v_i$. The polynomial fg has degree at most $\ell - k - 1$. Hence, we get ℓ equations (for each of $i = 1, \ldots, \ell$) in ℓ variables (the variables are the coefficients of the polynomials fg and g, where the leading coefficient of g is 1). Let f and f be a solution where f is a non-zero polynomial: f is a polynomial of degree at most f is a po

This completes the description of the tracing algorithm. Our tracing algorithm satisfies several properties:

- Full tracing Given a pirated key \bar{d} the tracing algorithm will recover all keys that were used in the construction of \bar{d} . In all previous tracing schemes only one pirate was guaranteed to be found. Note that the set of traced pirates may be a subset of the guilty coalition, i.e., the coalition may have used only a subset of the keys at its disposal to create \bar{d} . All keys that were actually used (i.e. the pirated key could not be constructed without them) will be found.
- Error free tracing The tracing algorithm is deterministic in the sense that there is no error probability. Any key output by the tracing algorithm must have participated in the construction of the pirated key.
- **Beyond threshold tracing** If more than k parties colluded to create the pirated key, then Berlekamp's algorithm may fail to recover the polynomial f, and tracing will fail. Above this bound, recent results of Guruswami and Sudan [7] may be used to output a list of candidate polynomials for f. The tracer gets a list of "leads" for the fraud investigation that includes the actual colluders. This will be effective against coalitions of size at most 2k-1.
- Running time The tracing algorithm requires that we solve a linear system of dimension ℓ (the total number of users). A naive implementation runs in time $O(\ell^2)$ (field operations). Asymptotically efficient versions of Berlekamp's algorithm run in time $\tilde{O}(\ell)$, where the "soft-Oh" notation hides polylog terms. The fastest known algorithm, due to Pan [12], runs in time $O(\ell \log \ell \log \log \ell)$.

4.2 Black box tracing

The question that remains is the following: given a pirate decryption box (or pirated decryption software) how does one recover the representation \bar{d} used by the box? It is conceivable that a pirate box could decrypt while the tracer is unable to extract any representation from it. If the pirate box is tamper-resistant, the tracer might be unable to open it. Even after seeing the decoding logic, the tracer might be unable to efficiently deduce from it a valid representation.

The above discussion shows that it is desirable to enable the tracer to extract a key used by the box simply by observing its behavior on a few chosen ciphertexts. In other words, the tracing algorithm may only use the pirated box as an oracle. Given this oracle the tracing algorithm must output one of the keys at the pirate's possession. Previous traitor tracing schemes support this kind of black box tracing [4, 10]. Our scheme supports two types of black box tracing techniques. The first is very efficient, but assumes the pirate is restricted in how it constructs the decryption box. The second works against an arbitrary pirate, but is less efficient.

Single-key pirates. A natural strategy for a pirate to build a pirate decryption box is to form a new representation \bar{d} and then create a box that decrypts using this representation. We call this a "single-key pirate" since only a single representation of y is embedded in the pirate decoder. We will show an efficient black box tracing algorithm that works against a single-key pirate. Note that when a single user attempts to construct a decoder that cannot be traced back to him he is essentially acting as a single-key pirate.

We model the single-key pirate's behavior as if he is divided into two distinct parties. The first party is given the k keys $\bar{d}_1,\ldots,\bar{d}_k$, and creates some new key \bar{d} . By Lemma 1 we know \bar{d} must be a convex combination of the k given keys. The first party then hands \bar{d} to the second party. The second party, seeing only \bar{d} and the public key, is free to implement the decryption box however he wants. For convenience, we refer to the second party as the "box-builder".

We show how the tracer can extract the representation $\bar{d}=(\delta_1,\ldots,\delta_{2k})$ from a decoder created by a single-key pirate. The basic idea is to observe the decoder's behavior on invalid ciphertexts, e.g., $\tilde{C}=\langle S, h_1^{z_1},\ldots,h_{2k}^{z_{2k}}\rangle$, where the non-constant vector \bar{z} is chosen by the tracer. This ciphertext is invalid since the h_i 's are raised to different powers. The next lemma shows that the pirate box cannot distinguish invalid ciphertexts from valid ciphertexts (assuming the difficulty of DDH in G_q). Hence, on input $\tilde{C}=\langle S,H_1,\ldots,H_{2k}\rangle$ it must respond with A where

$$A = S/\prod H_i^{\delta_i} = S/\prod h_i^{z_i\delta_i}$$

Lemma 2. Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be a public key and let $\bar{d} = (\delta_1, \ldots, \delta_{2k})$ be a representation of y. Suppose that given \bar{d} and the public key the box-builder is able to construct a pirate decoder that will correctly decrypt all valid ciphertexts, but when given a random invalid ciphertext $\tilde{C} = \langle S, h_1^{z_1}, \ldots, h_{2k}^{z_{2k}} \rangle$ will output a value different from $S/\prod h_i^{z_i\delta_i}$, with non-negligible probability (over the choice of S and \bar{z}). Then the box-builder can be used to solve DDH in G_q .

Proof Sketch. Given a challenge tuple $\langle g_1,g_2,u,v\rangle$ we decide whether it is a random tuple or a Diffie-Hellman tuple as follows: build a public key $\langle y,h_1,\ldots,h_{2k}\rangle$ by picking random $a_i,b_i\in\mathbb{Z}_q$ for $i=1,\ldots,2k$ and setting $h_i=g_1^{a_i}g_2^{b_i}$. The element $y\in G_q$ is constructed as in the key generation algorithm, i.e. $y=\prod h_i^{\alpha_i}$ for random $\alpha_i\in\mathbb{Z}_q$. Next, build a ciphertext

$$\tilde{C} = \langle S, u^{a_1}v^{b_1}, \dots, u^{a_{2k}}v^{b_{2k}} \rangle$$

where S is random in G_q . Observe that if the challenge $\langle g_1, g_2, u, v \rangle$ is a Diffie-Hellman tuple then \tilde{C} is a random valid ciphertext. Otherwise, \tilde{C} is a random invalid ciphertext. Next, we ask the box-builder to build a pirate decoder given $(\alpha_1, \ldots, \alpha_{2k})$ and the public key. We then feed \tilde{C} into the pirate decoder. Since the decoder behaves differently for valid and invalid ciphertexts the result enables us to solve the given DDH challenge.

By querying at invalid ciphertexts the tracer learns the value $\prod h_i^{z_i\delta_i} = S/A$ for vectors \bar{z} of its choice. After 2k queries with random linearly independent \bar{z} , the tracer can solve for $h_1^{\delta_1}, \ldots, h_{2k}^{\delta_{2k}}$. Since the tracer knows the discrete log of the h_i 's base g (recall Step 2 of key generation) it can compute $g^{\delta_1}, \ldots, g^{\delta_{2k}}$. Ideally, we would like to use homomorphic properties of the discrete log to run the tracing algorithm of the previous section "in the exponents". Unfortunately, it is an open problem to run Berlekamp's algorithm this way. Instead, we can recover the vector $\bar{d} = (\delta_1, \ldots, \delta_{2k})$ from $\langle g^{\delta_1}, \ldots, g^{\delta_{2k}} \rangle$ by using recent results on trapdoors of the discrete log [11,13] modulo p^2q and modulo N^2 . For instance, the trapdoor designed by Paillier [13] shows that if encryption is done in the group $\mathbb{Z}_{N^2}^*$ then the secret factorization of N (known to the tracer only) enables the tracer to recover $\langle \delta_1, \ldots, \delta_{2k} \rangle$ mod N from $\langle g^{\delta_1}, \ldots, g^{\delta_{2k}} \rangle$. The tracing algorithm of the previous section can now be used to recover the keys at the pirate's possession. This completes the description of the black box tracing algorithm for single-key pirates.

Arbitrary pirates. Unfortunately, pirates do not have to limit themselves to a single-key strategy. For instance, the pirate could embed multiple representations of y in the pirate decoder. The decoder could use a different random convex combination of its representations each time it decrypts. This would defeat the approach described above. It is an open problem to build an extractor that will efficiently extract some representation in the convex hull of the keys given to the pirate.

To achieve black box tracing against an arbitrary pirate we rely on "black box confirmation". Suppose the tracer is given an arbitrary pirate decoder, and the tracer suspects a particular set T of at most k traitors. By querying the pirate decoder in a black box fashion, the tracer will be able to efficiently verify this suspicion with high probability. More precisely, the tracer will be able to determine that the pirate must possess some unknown subset of the keys of members of T.

Let $\bar{d}_1,\ldots,\bar{d}_k$ be the keys belonging to members of T and let g be a generator of G_q . To confirm its suspicion of T the tracer queries the decoder with an invalid ciphertext $\tilde{C}=\langle S,g^{z_1},\ldots,g^{z_{2k}}\rangle$, where the vector \bar{z} satisfies $\bar{z}\cdot\bar{d}_i=w$ for all $i\in T$. Here w is a random element of G_q . As in Lemma 2, the decoder cannot distinguish this invalid ciphertext from a real one. Consequently, it will respond with $A=S/\prod g^{z_i\delta_i}$ where $\langle \delta_1,\ldots,\delta_{2k}\rangle$ is some representation of y. If T is indeed the coalition that created the decoder, then by Lemma 1 we know that $\langle \delta_1,\ldots,\delta_{2k}\rangle$ is in the convex hull of $\bar{d}_1,\ldots,\bar{d}_k$. Hence, if T is the guilty coalition we know that $A=S/g^w$. Confidence in this test can be increased by making multiple queries, where each query is constructed independently using different S,\bar{z},w . If for a suspect coalition T the pirate decoder always responds with $A=S/g^w$ then the pirate must possess a subset of the keys belonging to T. Note that this confirmation algorithm does not require trapdoors of the discrete log.

Since we are able to do black box confirmation, we can do black box tracing by running the confirmation algorithm on all $\binom{n}{k}$ candidate coalitions (n is the total number of users in the system). This results in an inefficient tracing algorithm, but it shows that black box tracing is possible in principle.

5 Chosen ciphertext security

In a typical scenario where our system is used it is desirable to defend against chosen ciphertext attacks. Fortunately, our scheme can be easily modified to be secure against adaptive attacks. The modification is similar to the approach used by Cramer and Shoup [5]. As in Section 3 we work in a group G_q of prime order q. For example, G_q could be a subgroup of order q of \mathbb{Z}_p^* for some prime p where q|p-1.

Key generation: Let g be a generator of G_q . Pick random $r_1, \ldots, r_{2k} \in \mathbb{Z}_q$ and set $h_i = g^{r_i}$ for $i = 1, \ldots, 2k$. Next, we pick random $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$ and $\alpha_1, \ldots, \alpha_{2k} \in \mathbb{Z}_q$ and compute

$$y = h_1^{\alpha_1} h_2^{\alpha_2} \cdots h_{2k}^{\alpha_{2k}}; \quad c = h_1^{x_1} h_2^{x_2}; \quad d = h_1^{y_1} h_2^{y_2}$$

The public key is $\langle y, c, d, h_1, \ldots, h_{2k} \rangle$. The private key is as in Section 3, but also includes $\langle x_1, x_2, y_1, y_2 \rangle$. Hence, user *i*'s private key is $\langle \theta_i, x_1, x_2, y_1, y_2 \rangle$.

Encryption: To encrypt a message $M \in G_q$ do the following: pick a random element $a \in \mathbb{Z}_q$, and compute

$$S = M \cdot y^a \; ; \quad H_1 = h_1^a \; , \; \dots \; , \; H_{2k} = h_{2k}^a$$

$$\nu = \mathcal{H}(S, H_1, \dots, H_{2k}) \; ; \quad v = c^a d^{a\nu}$$

where \mathcal{H} is a collision resistant hash function (or chosen from a family of universal one-way hash functions). Set the ciphertext C to be

$$C = \langle S, H_1, \dots, H_{2k}, v \rangle$$

It is a bit surprising that the system can be made secure against chosen ciphertext attacks by appending a single element v to the ciphertext.

Decryption: To decrypt a ciphertext $C = \langle S, H_1, \dots, H_{2k}, v \rangle$ using a private key $\langle \theta_i, x_1, x_2, y_1, y_2 \rangle$ first compute $\nu = \mathcal{H}(S, H_1, \dots, H_{2k})$ and check that

$$H_1^{x_1 + y_1 \nu} \cdot H_2^{x_2 + y_2 \nu} = v$$

If the test fails, reject the ciphertext. Otherwise, output

$$M = S/U^{\theta_i}$$
 where $U = \prod_{j=1}^{2k} H_j^{\gamma_j}$

and $\gamma^{(i)} = (\gamma_1, \dots, \gamma_{2k}) \in \Gamma$ is the codeword from which θ_i is derived.

Tracing: The tracing algorithm remains unchanged.

We show that the scheme is secure against adaptive chosen ciphertext attack. In other words, we show that the scheme is secure in the following environment: an adversary is given the public key. It generates two messages M_0, M_1 and is given the encryption $C = E(M_b)$ for $b \in \{0,1\}$ chosen at random. The adversary's goal is to predict b. To do so he is allowed to interact with a decryption oracle that will decrypt any valid ciphertext other than C. If the adversary's guess for b is b' and the probability that b = b' is $\frac{1}{2} + \epsilon$ then we say that the adversary has advantage ϵ . The system is said to be secure against an adaptive chosen ciphertext attack if the adversary's advantage in predicting b is negligible (as a function of the security parameter).

Theorem 2. The above cryptosystem is secure against an adaptive chosen ciphertext attack assuming that (1) the Decision Diffie-Hellman problem is hard in the group G_q , and (2) the hash function \mathcal{H} is collision resistant (or chosen from a family of universal one-way hash functions).

We assume the hash function \mathcal{H} is collision resistant. Suppose there exists a polynomial time adversary \mathcal{A} that is able to obtain a non-negligible advantage in predicting b when the above cryptosystem is used. We show that \mathcal{A} can be used to solve the Decision Diffie-Hellman problem in G_q .

Given a tuple $\langle g_1, g_2, u_1, u_2 \rangle$ in G_q we perform the following steps to determine if it is a random tuple (i.e chosen from R) or a Diffie-Hellman tuple (i.e chosen from D):

Init Set $h_1 = g_1$ and $h_2 = g_2$. pick random $r_3, \ldots, r_{2k} \in \mathbb{Z}_q$ and set $h_i = g_2^{r_i}$ for $i = 3, \ldots, 2k$. Next, choose random $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$ and $\alpha_1, \ldots, \alpha_{2k} \in \mathbb{Z}_q$ and compute

$$y = h_1^{\alpha_1} \cdots h_{2k}^{\alpha_{2k}}; \quad c = h_1^{x_1} h_2^{x_2}; \quad d = h_1^{y_1} h_2^{y_2}$$

Challenge The adversary \mathcal{A} is given the public key and outputs two messages $M_0, M_1 \in G_q$. We pick a random $b \in \{0, 1\}$ and compute:

$$S = M_b \cdot u_1^{\alpha_1} u_2^{\alpha_2} \prod_{i=3}^{2k} u_2^{\alpha_i r_i}$$

$$H_1 = u_1 , \quad H_2 = u_2 , \quad H_3 = u_2^{r_3} , \quad \dots , \quad H_{2k} = u_2^{r_{2k}}$$

$$\nu = \mathcal{H}(S, H_1, \dots, H_{2k}) ; \qquad v = u_1^{x_1 + y_1 \nu} u_2^{x_2 + y_2 \nu}$$

The challenge ciphertext given to \mathcal{A} is $C = \langle S, H_1, \dots, H_{2k}, v \rangle$. **Interaction** When the adversary \mathcal{A} asks to decrypt a ciphertext

$$C' = \langle S', H'_1, \dots, H'_{2k}, v' \rangle$$

we respond as in a normal decryption: first we check validity of the ciphertext and reject invalid ciphertexts. For valid ciphertext we give \mathcal{A} the plaintext $M = S' / \prod_{j=1}^{2k} (H'_i)^{\alpha_j}$.

Output Eventually the adversary \mathcal{A} outputs a $b' \in \{0,1\}$. If b = b' we say the input tuple is from D otherwise we say R.

This completes the description of the algorithm for deciding DDH using \mathcal{A} . To complete the proof of Theorem 2 it remains to show two things:

- When $\langle g_1, g_2, u_1, u_2 \rangle$ is chosen from D the joint distribution of the adversary's view and the bit b is statistically indistinguishable from the actual attack.
- When $\langle g_1, g_2, u_1, u_2 \rangle$ is chosen from R the hidden bit b is (essentially) independent of the adversary's view.

The proofs of both statements are similar to the proofs given by Cramer and Shoup [5] and will be given in the full version of the paper. Based on the two statements a standard argument shows that if the adversary \mathcal{A} has advantage ϵ in predicting b then the above algorithm for deciding DDH also has advantage ϵ . This completes the proof of Theorem 2.

Key extraction and black box tracing. It is surprising that although the scheme is resistant to chosen ciphertext attack, the decryption box will decrypt invalid ciphertexts. In particular, it will decrypt an invalid ciphertext $\tilde{C} = \langle S, H_1, \ldots, H_{2k}, v \rangle$ where

$$S = M \cdot y^a$$
; $H_1 = h_1^a$, $H_2 = h_2^a$, $H_3 = h_3^{b_3}$, $H_4 = h_4^{b_4}$,..., $H_{2k} = h_{2k}^{b_{2k}}$
 $\nu = \mathcal{H}(S, H_1, \dots, H_{2k})$; $v = c^a d^{a\nu}$

This is an invalid ciphertext since the h_i 's are raised to different powers. It passes the decryptor's test since h_1 and h_2 are raised to the same power. It cannot be distinguished from a valid ciphertext, assuming the hardness of DDH in G_q . The ideas of Section 4.2 can then be applied for black box tracing in this setting.

6 Pitfalls in designing public key traitor tracing schemes

The pirate decoder need not work in the same way as a legitimate decoder in a traitor tracing scheme. For example, the pirate may be able to derive a short string w that enables decryption but is not a legitimate decryption key. If w could be derived from any subset of k private keys, then it would be impossible to trace. This successful pirate strategy can be shown to defeat one of the traitor tracing schemes proposed by Kurosawa and Desmedt in [8]. Their scheme works as follows:

Key generation: Let g be a generator of a group G_q of order q for some prime q. Choose a random polynomial $f(x) = a_0 + a_1x + \ldots + a_kx^k$ over \mathbb{Z}_q . Compute

$$y_0 = g^{a_0}, \dots, y_k = g^{a_k}$$

The public key is $\langle g, y_0, \dots, y_k \rangle$. The private key for user i is $d_i = f(i)$. **Encryption:** To encrypt a message $M \in G_q$ compute

$$C = \langle g^r, M \cdot y_0^r, y_1^r, \dots, y_k^r \rangle$$

where r is random in \mathbb{Z}_q .

Decryption: To decrypt a ciphertext $C = \langle a, b_0, \dots, b_k \rangle$ using user i's private key d_i compute:

$$\left(b_0 \cdot b_1^i \cdot b_2^{(i^2)} \cdots b_k^{(i^k)}\right) / a^{f(i)} = M$$

The authors show that given the public key and k private keys $f(i_1), \ldots, f(i_k)$ it is impossible to construct another private key f(j) for user j, unless the discrete log problem in G_q is easy. However, let $w = (u, w_0, w_1, \ldots, w_k)$ be any convex combination of the vectors v_1, \ldots, v_k , defined by:

$$v_1 = \left[f(i_1), \ 1, \ i_1, \ i_1^2, \dots, i_1^k \right]$$

$$\vdots \qquad \qquad \vdots$$

$$v_k = \left[f(i_k), \ 1, \ i_k, \ i_k^2, \dots, i_k^k \right]$$

Then w is not a legitimate private key, but it can be used to decrypt any ciphertext $C = \langle a, b_0, \dots, b_k \rangle$ since

$$b_0^{w_0} \cdots b_k^{w_k} / a^u = M$$

One can show that many of the convex combinations w cannot be traced to any traitor. To do so, one shows the existence of disjoint coalitions of size k that can all create the same w. This example illustrates the importance of black box tracing, to ensure that it is possible to trace no matter how the decoder is implemented. By increasing the degree of f from k to 2k, it should be possible to demonstrate k-resilient black box confirmation (and thus inefficient black box tracing) for this scheme as in Section 4.2.

7 Conclusion

We present an efficient public key solution to the traitor tracing problem. Our construction is based on Reed-Solomon codes and the representation problem for discrete logs. Traceability follows from the hardness of discrete log. The semantic security of the encryption scheme against a passive attack follows from the Decision Diffie-Hellman assumption. A simple extension achieves security against an adaptive chosen ciphertext attack under the same hardness assumption. The private key in all cases is just a single element of a finite field and can be as short as 160 bits. The cryptosystem can be made to work in any group in which the Decision Diffie-Hellman problem is hard. It is an interesting open question to improve on the "black box" traceability of our approach. Also, it seems reasonable to believe that there exists an efficient public key tracing traitors scheme that is completely collusion resistant. In such a scheme, any number of private keys cannot be combined to form a new key. Similarly, the complexity of encryption and decryption is independent of the size of the coalition under the pirate's control. An efficient construction for such a scheme will provide a useful solution to the public key tracing traitors problem.

To conclude, we mention an application of our system to defending against software piracy. Typically, when new software is installed from a CD-ROM the user is asked to enter a short unique key printed on the CD cover. This key identifies the installed copy. Clearly the key printed on the CD cover has to be short (say under 20 characters) since it is typed in manually. Our system can be used in this settings as follows: since our private key can be made 120 bits long (to achieve 2^{60} security) it can be printed on the CD cover (each character encodes 6 bits). The software on the CD is encrypted using our system's public key. When the user types in his unique CD key the software is decrypted and installed on the user's machine. However, if a software pirate attempts to create illegal copies of the distribution CD (say using a CD-ROM burner) he must also attach a short printed key to the disk. Using our system, the key he attaches to the bootlegged copies can be traced back to him.

Acknowledgments

The second author would like to thank his colleagues at Xerox PARC for helpful discussions.

References

- M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: The case of hashing and signing", in proc. Crypto '94, 216–233.
- L. Welch and E. Berlekamp, "Error correction of algebraic block codes", U.S. Patent No. 4,633,470, issued December 1986.
- D. Boneh, "The decision Diffie-Hellman problem", In proc. of the Third Algorithmic Number Theory Symposium (ANTS), LNCS, Vol. 1423, Springer-Verlag, pp. 48–63, 1998.

- 4. B. Chor, A. Fiat and M. Naor, "Tracing traitors", in proc. Crypto '94, pp. 257-270.
- 5. R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack", in proc. Crypto '98, pp. 13–25.
- C. Dwork, J. Lotspiech and M. Naor, "Digital signets: self-enforcing protection of digital information", in proc. of STOC '96, pp. 489-498, 1996.
- V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraicgeometric codes" in proc. 39th IEEE Symposium on Foundations of Computer Science (FOCS), 1998.
- K. Kurosawa, and Y. Desmedt, "Optimum traitor tracing and asymmetric schemes", in proc. of Eurocrypt '98, pp. 145–157.
- A. Menezes, P. van Oorschot and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.
- M. Naor and B. Pinkas, "Threshold traitor tracing", in proc. Crypto '98, pp. 502–517.
- T. Okamoto, S. Uchiyama, "A new public key cryptosystem as secure as factoring", in proc. Eurocrypt '98, pp. 308-318.
- V. Pan, "Faster solution of the key equation for decoding BCH error-correcting codes" in proc. 29th ACM Symposium on Theory of Computation, pp. 168–175, 1997.
- 13. P. Paillier, "Public-Key Cryptosystems Based on Discrete Logarithm Residues", in proc. Eurocrypt '99, pp. 223–238.
- B. Pfitzmann, "Trials of traced traitors", in proc. Information Hiding Workshop, 49-64, 1996.
- B. Pfitzmann and M. Waidner, "Asymmetric fingerprinting for larger collusions", in proc. ACM Conference on Computer and Communication Security, 151–160, 1997.
- 16. D. Stinson and R. Wei, "Combinatorial properties and constructions of traceability schemes and frameproof codes", SIAM Journal on Discrete Math, 11(1), 41–53, 1998.

Dynamic Traitor Tracing

Amos $Fiat^{1,2}$ and $Tamir Tassa^2$

Abstract. Traitor tracing schemes were introduced so as to combat the typical piracy scenario whereby pirate decoders (or access control smartcards) are manufactured and sold by pirates to illegal subscribers. Those traitor tracing schemes, however, are ineffective for the currently less common scenario where a pirate publishes the periodical access control keys on the Internet or, alternatively, simply rebroadcasts the content via an independent pirate network. This new piracy scenario may become especially attractive (to pirates) in the context of broadband multicast over the Internet. In this paper we consider the consequences of this type of piracy and offer countermeasures. We introduce the concept of dynamic traitor tracing which is a practical and efficient tool to combat this type of piracy. We also consider the static watermarking problem, presented by Boneh and Shaw, and derive bounds on the performance parameters of the "natural majority algorithm".

1 Introduction

The subject of this paper is protecting ownership rights of intellectual property. The best example is that of pay TV systems where subscribers may access specific channels or programs by purchasing their viewing rights. In such systems, the content is distributed via terrestrial, cable or satellite broadcast and, hence, a conditional access system must be utilized in order to guarantee that only paying subscribers can access the content for which they have paid. But even though pay TV systems are the most outstanding realization of our model, there are others as well: conditional access systems are also used to protect pay services on The Web.

In this paper we address the issue of protecting ownership rights against piracy whereby unauthorized users get access to the content. Pirates make a business of breaking the security safeguards of the conditional access system and sell devices that allow unauthorized users to view the content illegally. To prevent such unauthorized access, cryptography is often used: the conditional access system makes use of secret keys in order to allow only legitimate users access to the content.

The use of tamper-resistant devices for conditional access systems is the norm, so as to prevent access to the underlying keys. However, recent advances

Department of Computer Science, School of Mathematical Sciences Tel Aviv University, Tel Aviv, Israel

² Algorithmic Research Ltd., 10 Nevatim St., Petah Tikva, Israel

in attacks on tamper resistant devices, most notably differential power analysis and timing attacks [6], have compromised unqualified reliance on tamper resistance. Thus, a more realistic model must assume that piracy will occur and, therefore, countermeasures should be taken once piracy has been observed. Such countermeasures should be capable of the following:

- Trace the source of piracy.
- Disconnect it and its dependent unauthorized users from further transmittal of information.
- Harming no legitimate users.
- Supply legal evidence of the pirate identity.

The traitor tracing schemes of Chor et al [2] adopt the following model: pirate decoders that allow access to the content may be manufactured but such decoders, if captured, must inherently contain identifying information that will allow the broadcaster to cut them off from future broadcasts. Additionally, the source of piracy can be detected and legal means can be taken.

To do so, Chor et al introduce a new form of cryptography that uses one encryption key and multiple distinct decryption keys, with the property that one cannot compute a new decryption key from a given set of keys. The traitor tracing schemes of [2,8,10] approximate such a scheme. Two cost measures are to be considered when implementing such schemes: storage requirements at the user end and the necessary increase of bandwidth.

The Achilles' heel of such traitor tracing schemes is their underlying assumption that pirates provide unauthorized subscribers with decoders capable of decoding the original broadcast. Such schemes would be ineffective if the pirate were simply to rebroadcast the original content using a pirate broadcast system.

This paper deals with the latter scenario: Even if the pirate rebroadcasts the original content to pirate users, countermeasures can be activated in order to trace and disconnect the so-called *traitors*, *i.e.*, the real subscribers controlled by the pirate.

To accomplish this, watermarking methods are implemented, allowing the broadcaster to generate different versions of the original content, with no noticeable degradation in the content quality. The schemes which we introduce and discuss here, use the watermarks found in the pirate copy to trace its supporting traitors. A fundamental assumption in this context is that it is possible to generate tamper-resistant watermarks that a pirate could not remove. Cox et al [3] have introduced methods to create such secure and robust watermarks.

Watermarking schemes were introduced and discussed by Boneh and Shaw in [1]. In their study they assumed that the content is watermarked once, prior to its broadcast. The schemes of [1] were designed to trace the source of piracy once a pirate copy of the content is captured. The traitor tracing schemes of [2] are similar in that sense: each decoder is personalized by a unique allocation of decryption keys, once, before it is sold to a subscriber. Only when a pirate decoder is captured, the traitor tracing schemes are activated in order to trace a legal

decoder used in building the pirate unit. Both the watermarking schemes of [1] and the traitor tracing schemes of [2] are probabilistic. Namely, the evidence they provide against the suspected traitor is accompanied by a small error probability (that can be made as small as desired). It should be noted that even though the watermarking codes of [1] and the traitor tracing key assignment tools of [2] have, seemingly, an entirely different motivation, traitor tracing schemes can be translated into watermarking codes as described in [1].

Like [1,2], we make use of marking codes but, unlike [1,2], our codes are generated on the fly. In our model, we use the feedback from the pirate distribution network in order to lock onto the traitors much more efficiently. We refer to this latter model as the dynamic model while the former one [1,2] is referred to as the static model. The dynamic model is very natural and has great practical applications in the context of protecting intellectual rights in broadcast systems. The static model, on the other hand, is suitable for electronic data distribution systems.

To understand the fundamental contribution of the dynamic model, we consider the following scenarios:

- 1. Dynamic schemes decide about the number of active traitors on the fly, based on the feedback from the pirate network, and adapt their behavior accordingly. That is impossible in the static model, where an a priori bound on the number of traitors is required (the lack of such a bound renders any static method completely unreliable).
- 2. Even if an a priori bound is known, but false incriminations of innocent users are strictly prohibited, there is an exponential performance improvement of dynamic methods over static ones. This exponential gap implies that static schemes are simply impossible in such settings.
- 3. If an a priori bound is known, and one allows a constant probability, $\varepsilon > 0$, of false incrimination, static schemes pay an additional $\log(1/\varepsilon)$ factor in performance that is not required by dynamic methods.

1.1 Organization of the Paper

The paper is organized as follows: in $\S 2$ we formalize the model, introduce the basic terminology and discuss relevant implementation issues. Then, in $\S 2.3$, we prove a fundamental result that connects the size of the marking alphabet to the number of active traitors. A byproduct of our analysis is that the probabilistic nature of the codes in [1] is inherent, *i.e.*, no code of that nature can avoid making errors.

§3 is devoted to the dynamic setting. Three deterministic algorithms are presented and compared: two of them have optimal spacial efficiency while the other one excels in temporal efficiency.

In §4 we discuss the static setting and study a scheme proposed in [2]. Using a more careful analysis, we are able to obtain performance estimates which are

much better than those obtained in [2]. We note that this scheme, with our improved estimates, may be combined with the codes and distribution mechanism of [1] in order to yield schemes with optimal spacial efficiency.

Finally, in §5, we list several interesting open problems that our study raises.

1.2 Related Work

The concepts of frameproof codes and secure codes were defined in [1]. Additional explicit constructions of frameproof codes were given in [9].

There are a variety of slightly different definitions of frameproof and secure codes. Generally, a frameproof code is an assignment of codewords to users so that no coalition whose size is no more than some preset limit p can "frame" an innocent user. A coalition of size p can compute new codewords from the set of codewords assigned to its members. The rules by which new codewords can be computed vary slightly from paper to paper. The different rules refer specifically to what is permissible when combining two or more codewords to create another. For example, given two codewords x and y that differ in their ith coordinate, $x_i \neq y_i$, one can generate a new codeword z for which either

- 1. $z_i \in \{x_i, y_i\}$ (as in the CFN-model [2], which coincides with ours), or,
- 2. z_i is either an arbitrary element of the underlying alphabet or something entirely unrecognizable (as in the BS-model [1,9]).

Rather than talk in terms of codewords, let us translate these two models to the watermarking terminology:

- Given two variants of a movie segment, v_1 and v_2 , if the only possible choice for the pirate is to transmit either v_1 or v_2 , then we are in the CFN-model.
- Given two variants of a movie segment, v_1 and v_2 ($v_1 \neq v_2$), if the pirate can produce any variant out of all possible variants, or something entirely unrecognizable, then we are in the BS-model [1,9].

We justify our choice of model below, but first a few words to avoid confusion. We use the term "the CFN-model" somewhat misleadingly because [2] does not deal with the watermarking problem at all. Rather, [2] deals with the assignment of keys to decoders so as to recognize the source of a pirated decoder. One of the properties of cryptographic keys is that given two different symmetric keys, it (usually) makes little sense to try to combine them in some way and obtain a meaningful third key. Thus, if the pirate has to choose between using key k_1 or key k_2 , he can choose either one of them, or none of them, but would not find it useful to use, say, $k_1 \oplus k_2$. In the translation between the traitor tracing schemes of [2] and watermarking schemes, the different keys are analogous to different variants of a segment, whence the term "the CFN-model" in the context of watermarking.

Given some variants of a movie segment, it would seem most unfeasible to compute a new valid variant. The reason for that is that in any reasonable watermarking scheme the pirate would not have the information essential to generate such a variant. It may be possible, however, to remove all watermarking information while paying the price of quality degradation. But even if that is possible, it would be difficult to do so, and the pirate would not necessarily know whether he was successful or not. This is why we find the CFN-model a more realistic model in this context. It should be noted that in our dynamic schemes, if we cannot recognize the variant that is currently transmitted by the pirate, we simply ignore the corresponding segment and wait for the next one. Even if the pirate is successful in removing the watermarking with probability q (the value of which is dictated by the technical difficulties, as well as by the need to have a rebroadcast with a reasonable quality), it implies a 1/(1-q) constant factor in convergence time.

Finally, from a practical perspective on the immediate future, we can justify our model for much the same reasons as in [2] (see §2.2).

In the static model, a related paper by Stinson and Wei [9] constructs frame-proof schemes as well as traceability schemes. In this context, traceability schemes coincide with simple majority deterministic tracing algorithms that are not allowed to make any error. In [9, Theorem 5.5] they give a bound that connects all the parameters of the problem: the number of users, the size of the coalition of traitors, the size of the marking alphabet and the length of the codewords. That bound may be translated into a lower bound on the length of codewords which is proportional to the number of traitors times the log of the number of users. We conjecture in this paper that the true lower bound is much higher and is in fact exponential in the number of users.

Other related work about traitor tracing may be found in [4,7,8].

2 The Model

In our model, content consists of multiple segments, e.g., a segment could be one minute worth of video. It is possible to generate multiple variants of each segment. Those variants must meet with the following two requirements:

- Similarity. Fundamentally, all variants carry the same information to the extent that humans cannot distinguish between them easily.
- **Robustness.** Given any set of variants, v_1, \ldots, v_k , it is impossible to generate another variant that cannot be traced back to one of the original variants, $v_i, 1 \le i \le k$.

Clearly, those requirements place an upper bound on the number of variants that can be generated from a single content segment (the reader is referred to [3] where methods to generate such watermarks are introduced). Content for which some or all of the segments have been assigned variants is called a watermarked content or a version.

In general, the watermarking problem is to generate multiple versions of watermarked content so that, given a black market copy of that content, the watermarks embedded in that copy would lead to the identification of its source.

A watermarking scheme for tracing traitors consists of two essential parts:

- 1. Watermark distribution: An algorithm that assigns each subscriber a watermarked copy of the content.
- 2. **Tracing and incrimination:** An algorithm that, given an illegal copy of the content, uses the watermarks embedded in it in order to trace back at least one of the traitors that participated in producing that copy.

A watermarking scheme is called *deterministic* if it traces and incriminates all traitors and no one else but the traitors. On the other hand, schemes in which there is a small chance of false incrimination are referred to as *probabilistic*.

The two key performance parameters in this context are r, the number of different variants used per segment and m, the number of content segments. One way to view our model of watermarking is that it is an embedding of a codeword in the content, where r is simply the size of the marking alphabet and m is the length of the codeword.

The following terminology is used throughout the rest of the paper:

- 1. **The center** is the source of the content and its watermarked copies.
- 2. The users, or subscribers, denoted by $U = \{u_1, \ldots, u_n\}$, are recipients of the content.
- 3. Some of the users may collude in order to distribute illegal copies of the content to pirate subscribers. We refer to such users as **traitors** and to their coalition as **the pirate** and denote them by $T = \{t_1, \ldots, t_p\}, T \subset U$.
- 4. The marking alphabet that is used to generate codewords is denoted by $\Sigma = {\sigma_1, \ldots, \sigma_r}$.
- 5. For a given segment $1 \leq j \leq m$ and a mark σ_k , $1 \leq k \leq r$, $S_k^j \subset U$ denotes the subset of subscribers that got variant σ_k of segment j.

We consider in this paper two settings: a dynamic setting and a static one.

The dynamic setting assumes on-line feedback from the pirate subscribers to the center. Such a scenario is feasible in cases like TV broadcast, where the pirate rebroadcasts the content, say, on the Internet. The center can therefore see the current pirate broadcast and adapt its watermark distribution in the next segments in order to trace the traitors efficiently.

In such a scenario, the number of variants that are transmitted simultaneously, r, is proportional to the bandwidth requirements, while m, the number of segments or search steps, is proportional to the time required to trace the traitors (the convergence time).

In **the static setting** there is a one time marking of the content per user. Only when a black market copy is found, the tracing and incrimination algorithm is activated. This model is suitable for, e.g., DVD movie protection. Obviously, performance in such a rigid setting with no on-line feedback is less efficient than that in the dynamic setting. This setting is also somewhat less useful than the dynamic setting because there are fewer effective countermeasures: legal action post-factum is the only recourse (as opposed to the dynamic setting that allows immediate disconnection of the traitors).

As in the dynamic setting, r and m are relevant performance measures, but they have a slightly different significance. Here, r determines the relative extra expense required for watermarking, while m is limited by the maximal number of segments that can fit into the given content.

2.1 Control Overhead

A key issue is to control what users get what variant of every segment. The simplest way to do so is as follows:

- 1. Every user has a unique symmetric key in common with the center.
- 2. Prior to every segment transmission, the center distributes keys to users, using individually encrypted transmissions: If user i is to get variant ℓ of segment j, then the center sends an individually encrypted transmission to user i containing key K_{ℓ}^{j} , where all such keys are generated at random.
- 3. The center now transmits multiple variants of the jth segment, where variant ℓ is encrypted under key K_{ℓ}^{j} .

The broadcast overhead for implementing such a scheme is composed of two components:

- 1. Before each segment, the center needs to transmit individual (short) messages to every user that contain the relevant keys.
- 2. The center needs to broadcast multiple variants of every segment; this is a high overhead component because it multiplies the total bandwidth by the number of different variants.

There are a number of mechanisms that allow us to reduce this overhead. First, rather than using individually encrypted messages we can use broadcast encryption schemes [5]. At first glance it seems that this creates a problem because broadcast encryption schemes require an a priori knowledge of the number of traitors, whereas we claim that we do not need to know this. However, we never kill off a suspect user unless we know for sure that he is a traitor. Hence, we can start with an estimate on the number of traitors, and if this estimate turns out to be wrong, we can simply restart with a higher initial estimate for the broadcast encryption component.

Next, we do not necessarily have to change keys between segments for all users. In fact, we only need to change keys in case that a set of users is split up into two or more subsets, or if we perform a union between sets of users. Thus, even if one uses the naive approach (individual transmissions to every user) it turns out that our 2p+1 algorithm, §3.3, only requires O(np) individual transmissions for all segments.

However, the more expensive overhead is in the simultaneous transmission of multiple variants of a segment. Here, one can make use of the nature of the problem to reduce bandwidth overhead. Even if, say, 90% of the movie were transmitted entirely in the clear (and not watermarked), while only the remaining 10% were to be watermarked and protected, this would create problems for

the pirate. A pirate copy that misses 10% of the movie is not very valuable. This means that we can transmit multiple variants for only a (relatively) small part of the movie, hence reducing the bandwidth overhead considerably.

2.2 Short-Term Practical Considerations

In the immediate future, it seems rather unlikely that the actual MPEG-II transmission will be rebroadcast over the Internet (due to lack of bandwidth). Thus, it may be that the setting described in this paper is not required in the immediate future. Hence, we briefly describe how to adapt our schemes for conditional access schemes used today.

All conditional access schemes today use rapidly changing symmetric keys to encrypt the content. These symmetric keys, known as "control words", are replaced (say, every 5 seconds) through the use of so-called "Entitlement Control Messages" (ECMs). An underlying hidden assumption in common to all these schemes is that the control words will not be retransmitted by the pirate to his subscribers.

This assumption is true if the bandwidth available to the pirate for retransmission is lower than that required to retransmit the control words. Thus, the center must set the control word change rate to reflect the bounds on the pirate transmission capabilities.

Nonetheless, the problem with this setting is that the pirate could still transmit the secret(s) used to obtain the control words from the ECMs.

Now, we can simply make use of dynamic traitor tracing schemes, where rather than watermarking multiple variants of the content, we encrypt the control words under several different keys (analogous to variants).

In this setting the control overhead is much lower (multiple ECM streams) and our model that disallows computation of a third variant from two existing variants is obviously justified.

2.3 Deterministic Lower Bound

Before discussing the dynamic and static settings separately, we state the following fundamental theorem which applies in both settings:

Theorem 1. If the pirate controls p traitors then:

- (a) There exists a deterministic watermarking scheme with $|\Sigma| = p + 1$.
- (b) No watermarking scheme that uses an alphabet of size $|\Sigma| \leq p$ can be deterministic.

In other words, a watermarking scheme must use an alphabet of size p + 1 at the least in order to trace and incriminate all traitors and no one but the traitors.

In the static setting, this requires to have an a priori bound on the number of traitors. However, as we shall see later, any deterministic scheme in this setting is bound to be impractical anyway and, therefore, only probabilistic schemes are considered.

In the dynamic setting, the scheme can learn on the fly what is the number of traitors and adapt its alphabet size accordingly; hence, no a priori information as to the number of traitors is required.

Proof. Here we prove part (b) of the theorem. As for the proof of part (a), see §3.2 and §3.4 where such schemes are described. Given some innocent subscriber of the system, $u \in U \setminus T$, we define $T_0 := T = \{t_1, \ldots, t_p\}$ and $T_i = T \cup \{u\} \setminus \{t_i\}$ for all $1 \leq i \leq p$. In addition, let us denote $T^* = T \cup \{u\}$. Now, assume that the pirate T adopts the following strategy: in segment j it rebroadcasts one of the variants σ_k for which $|S_k^j \cap T^*| \geq 2$. The existence of such a variant is guaranteed by the pigeon hole principle, since $|\Sigma| < |T^*|$. Clearly, the chosen subset S_k^j intersects T_i for all $0 \leq i \leq p$. Hence, it is impossible to distinguish between the real coalition of traitors, T_0 , and the camouflage sets T_i , $1 \leq i \leq p$. Therefore, the scheme could never point out the p true traitors out of the p+1 subscribers in T^* .

We would like to point out that Theorem 1 is a generalization of [1, Theorem 4.2] which was restricted to the case p=2. In addition, we proved this lower bound on the alphabet size under the more general assumption of robustness of the watermarks (whereas the proof of [1, Theorem 4.2] relied on the ability of the traitors to destroy marks).

3 The Dynamic Setting

3.1 Preliminaries

In the dynamic scenario, the pirate T broadcasts at every time segment $j, j \geq 1$, one of the variants owned by the traitors controlled by him, $t_i, 1 \leq i \leq p$. Let us denote that variant by s_j and denote by B_j the pirate transmission up to time j, $B_j = (s_1, \ldots, s_j) \in \Sigma^j$ (those are available, say, by registering as a pirate user).

The goal of the watermarking scheme is to disconnect all subscribers in T, thus rendering the pirate inoperative. Additionally, it would be bad to disconnect innocent subscribers $u \in U \setminus T$. Hence, only deterministic schemes are considered in this case.

Formally, a dynamic watermarking scheme is a function $f: U \times \Sigma^* \mapsto \Sigma \cup \{\sigma_0\}$. For all $j \geq 1$, f induces a partition of U into the disjoint sets $S_k^j = \{u \in U: f(u, B_{j-1}) = \sigma_k\}, 0 \leq k \leq r$. This is interpreted as follows:

- 1. At time $j \ge 1$, users $u \in S_k^j$, $k \ge 1$, get variant σ_k of content segment j.
- 2. At time $j \geq 1$, users $u \in S_0^j$ are disconnected, i.e., get no variant of content segment j. We assume that $S_0^j \subset S_0^{j+1}$ for all $j \geq 1$, i.e., disconnection is permanent.

In the following subsections we describe several deterministic schemes and study their performance in terms of r – the number of variants that they require in each segment, and m – the number of steps required to trace and disconnect all traitors. Those schemes do not require any a priori knowledge of p; instead, each of those schemes keeps track of a lower bound on the number of traitors. That value is initially set to zero and only when piracy is detected the scheme increases it to one. That lower bound is increased only when the findings of the scheme up to that point imply that this is valid. That lower bound is denoted by t in the first two schemes §3.2-3.3. In the third scheme §3.4, another related parameter appears and t has there a slightly different interpretation.

3.2 First Scheme: r = p + 1, Impractical Convergence Time

The following straightforward scheme makes use of (no more than) p+1 variants in each segment. Therefore, it has an optimal spacial efficiency. However, its temporal efficiency is very bad as its convergence time is exponential in n.

- 1. Set t = 0.
- 2. Repeat forever:
 - (a) For all selections of t users out of U, $\{w_1, \ldots, w_t\} \subset U$, produce t+1 distinct variants of the current segment and transmit the ith variant to w_i , $1 \le i \le t$, and the (t+1)th variant to all other users, until the pirate transmits a recognized variant.
 - (b) If the pirate ever transmits variant i for some $i \leq t$, disconnect the single user w_i and decrement t by one. Otherwise, increment t by one.

Clearly, this algorithm will trace and disconnect all traitors, t_i , $1 \le i \le p$, because when t reaches the value of p, one of the selections will be that in which $w_i = t_i$, $1 \le i \le p$; when that selection is made, either piracy stops or one of the traitors will incriminate himself. The convergence time for this algorithm, though, may be as large as $\binom{n}{p} + 2 \cdot \sum_{t=0}^{p-1} \binom{n}{t}$, hence it is impractical.

3.3 Second Scheme: r = 2p + 1, Efficient Convergence

Next, we present an algorithm that requires 2p+1 keys but removes all traitors within $O(p \log n)$ steps. We note that any binary decision tree for determining all p traitors within a user group of size n has a depth of $p \log n$, as implied from the information theoretic bound.

Throughout this algorithm, the set of subscribers, U, is partitioned into 2t+1 subsets, $U = \bigcup_{S \in P} S$, where $P = \{L_1, R_1, \ldots, L_t, R_t, I\}$, and each of those sets receives a unique variant. Hence, there are never more than 2t+1 simultaneous variants; since t – the lower bound on the number of traitors – never exceeds p – the true number of traitors, the upper bound on the size of the alphabet, $|\Sigma| \leq 2p+1$, is respected.

An invariant of the algorithm is that the union $L_i \cup R_i$ contains at least one traitor for all $1 \le i \le t$. I is the complementary subset of users that is not known to include a traitor.

- 1. Set t = 0, I = U, $P = \{I\}$.
- 2. Repeat forever:
 - (a) Transmit a different variant for every non-empty set of users $S \in P$.
 - (b) If the pirate transmits a variant v of the current segment then:
 - If v is associated with I, increment t by one, split I into two equal sized subsets, L_t and R_t , add those sets to P and set $I = \emptyset$.
 - If v is associated with one of the sets L_i , $1 \le i \le t$, do as follows:
 - i. Add the elements in R_i to the set I.
 - ii. If L_i is a singleton set, disconnect the single traitor in L_i from the user set U, decrement t by one, remove R_i and L_i from P and renumber the remaining R_i and L_i sets in P.
 - iii. Otherwise (L_i is not a singleton set), split L_i into two equal sized sets, giving new sets L_i and R_i .
 - If v is associated with one of the sets R_i , $1 \le i \le t$, do as above while switching the roles of R_i and L_i .

Theorem 2. The watermarking scheme which the above algorithm implements, traces all p traitors within $m = p \log n + p$ time steps, while using no more than r = 2p + 1 simultaneous variants.

Proof. It is clear that at any given stage, the union $L_i \cup R_i$, $1 \le i \le t$, contains at least one traitor (this is an invariant of the algorithm). Hence, the number of $\{L_i, R_i\}$ pairs, t, cannot exceed the total number of traitors, p. Since the scheme uses at each stage no more than 2t + 1 variants, the upper bound of 2p + 1 simultaneous variants is respected.

As for the convergence time, consider a sequence of tracing steps through which a traitor is isolated in successively smaller subsets, L_i or R_i . Clearly, each single traitor will be isolated within $\log n$ steps. Hence, all traitors will be isolated within $p \log n$ steps. Once all traitors are isolated, the pirate's broadcast must incriminate them all after additional p steps.

3.4 Third Scheme: r = p + 1, Improved Convergence Time

Here, we present another algorithm that uses an optimal alphabet of size p+1. Its convergence time is bounded by $O(3^p p \log n)$ which is a dramatic improvement over $\binom{n}{p} \approx n^p$ of the scheme in §3.2, though still non-polynomial in p. Our new algorithm is very similar to the previous one, §3.3, in the sense that the partitions that it uses are of the same form, $U = \bigcup_{S \in P} S$, $P = \{L_1, R_1, \ldots, L_t, R_t, I\}$, and it has the same invariants: the union $L_i \cup R_i$ contains at least one traitor for all $1 \leq i \leq t$, while I is not known to include any traitor.

The difference between the two algorithms (which is manifested most notably in their running time) stems from the fact that we may not have sufficient variants for all the 2t+1 sets in P (due to the tighter restriction on the simultaneous number of variants). Hence, if in the previous algorithm we had only one dynamic parameter, t, that indicated both the number of $\{L_i, R_i\}$ pairs and the current lower bound on the number of traitors, in this algorithm there are two dynamic parameters:

- 1. k, the current lower bound on the number of traitors (i.e., how many traitors are known to exist at this stage of the search), and
- 2. t, the number of pairs of subsets, $\{L_i, R_i\}$, in the partition P of U. Each of those pairs is known to include at least one traitor.

Clearly, $k \geq t$; later, we shall see that $k \leq 2t$. Hence, the knowledge that the tracing scheme holds in each step may be summarized as follows:

$$|T| \ge k$$
 and $|(L_i \cup R_i) \cap T| \ge 1$ $1 \le i \le t$.

In the 2p+1 algorithm, having the luxury of assigning a unique variant to each set $S \in P$, we were guaranteed to make a progress in every step, where a progress means the splitting of one of the sets in P towards closing on the traitor(s) in that set. Here, however, we cannot do so since we are limited to use no more than r = k+1 different variants in each step. Hence, instead of achieving a progress in each step, the algorithm that we present below is guaranteed to achieve a progress within a finite number of steps.

- 1. Set t = 0, k = 0, I = U, $P = \{I\}$.
- 2. Repeat forever:
 - (a) For every selection of $\{S_1, \ldots, S_k\} \subset P$ where $S_i \in \{L_i, R_i\}$, $1 \le i \le t$, and S_i , $t+1 \le i \le k$ are any other k-t sets from P, produce k+1 variants, σ_i , $1 \le i \le k+1$. Transmit σ_i to S_i for all $1 \le i \le k$, while all remaining users get variant σ_{k+1} .
 - (b) Assume that the pirate transmits at some step a variant σ_i that corresponds to a single set in P (when k < 2t those are the variants σ_i where $1 \le i \le k$; when k = 2t, on the other hand, all variants correspond to a single set, since then also σ_{k+1} is transmitted to just one set). In that case:
 - i. If σ_i corresponds to an L_i set, then that set must contain a traitor. In that case we add the corresponding complementary set, R_i , to I and split L_i into two equal sized sets giving a new $\{L_i, R_i\}$ pair. In this case neither t nor k changes but eventually, when the size of the incriminated set is one, we may disconnect the traitor in that set. When this happens, we restart the loop after decrementing k and t by one.
 - ii. If σ_i corresponds to an R_i set, we act similarly.
 - iii. If σ_i corresponds to I, it allows us to increment t by one (and k as well, if k was equal to t), split I into a new $\{L_t, R_t\}$ pair, set $I = \emptyset$ and restart the loop.
 - (c) If k < 2t and the pirate always transmits σ_{k+1} , then after completing the entire loop we may increment k by one and then restart the loop.

Given k and t, the basic loop consists of $2^{2t-k+1}\binom{t}{k-t-1}+2^{2t-k}\binom{t}{k-t}$ rounds. Since, in the worst case, we may need to repeat the loop from k=t to k=2t until we split a set, we are guaranteed to make a progress in the form of splitting a set within no more than $2 \cdot 3^t - 1$ steps (which equals the sum over k of the

above terms). This is always bounded by $2 \cdot 3^p$. Hence, convergence is guaranteed within no more than $(2 \cdot 3^p p \log n + p)$ steps. This bound is not tight, but, on the other hand, it is clear that any upper bound on the convergence time cannot be less than $O(2^p p \log n)$ rounds. Hence, this algorithm is exponential in p.

Note that this algorithm actually combines the two previous ones. It uses the same search tree as the 2p+1 algorithm of §3.3. However, when a gap is created between k and t, the previous p+1 algorithm of §3.2 is implemented in order to trace the additional k-t subsets of P that contain a traitor. We could, of-course, avoid the inefficient algorithm of §3.2 and, instead, implement again the algorithm of §3.3 in a recursive manner. However, that would make the algorithm quite intricate, while not improving its convergence time substantially.

4 The Static Setting

4.1 Preliminaries

In this scenario, the content is marked once, by dividing it into m segments and marking each segment using the marking alphabet Σ , $|\Sigma| = r$. Each user is given a unique watermarked copy (version) of the content that is randomly chosen out of the r^m possible versions. The following notations and assumptions are used throughout this section:

- C(u) is the copy, or version, that user $u \in U$ got. Namely, C is a one-to-one random function from U to Σ^m , which is assumed to be uniformly distributed.
- C(T) is the closure of $\{C(t_1), ..., C(t_p)\}$; i.e., all vectors $C \in \Sigma^m$ for which $C_j \in \{C(t)_j : t \in T\}$ for all j = 1, ..., m. In other words, C(T) consists of all versions that can be produced by T from the p versions of its members.
- $-C^*$ stands for the copy that T produced and distributed in the black market.
- For any $x, y \in \Sigma^m$, $M(x, y) = \frac{1}{m} \cdot |\{j : x_j = y_j, 1 \le j \le m\}|$ denotes their matching score.

We consider the natural majority algorithm: given a black market copy, C^* , the tracer incriminates and punishes that user u who has a maximal matching score, $M(C(u), C^*)$. This is the same algorithm that was suggested and studied in [2]. We carry out here a more careful analysis that enables us to improve the performance estimates that were obtained in [2].

Note that this scheme is the static analogue to the dynamic schemes described in the previous section, because all we care about are exact "matches" and the information on the matches is taken in each segment separately (as opposed to, e.g., the Boneh-Shaw scheme [1] where the matching information is considered in the context of groups of segments).

The best strategy for the pirate against the majority algorithm is to select that codeword $C^* \in C(T)$ that minimizes $\max_{t \in T} M(C(t), C^*)$. However, finding the exact solution for this problem seems to be a difficult task. A random choice of C^* seems like a more practical solution for the pirate. We will assume that

 C^* is uniformly distributed amongst all vectors in C(T), i.e., for each segment j, one of the available variants, $\{C(t)_j: t \in T\}$, is chosen uniformly at random (without repetitions).

Summary of results. Hereinafter we assume that U, T and Σ are given (where $n > p \ge 1$ and $r \ge 2$), as well as a small parameter $\varepsilon > 0$. We set the probability space to be that which consists of all possible tracer's allocations of codewords, $C(\cdot)$, and pirate's selections of $C^* \in C(T)$ with uniform distributions. The question which arises is as follows: Can we find a lower bound M such that the above described scheme, when using m > M segments, would incriminate a true traitor in probability of at least $1 - \varepsilon$? In Theorem 3 we provide a positive answer to this question.

Next, we provide a simple algorithm to compute a lower bound M_1 such that the scheme would perform as desired for all $m \geq M_1$. This lower bound is significantly better than the lower bound obtained in [2].

We also study the asymptotic behavior of the lower bound for m in the two regimes $r \ge p+1$ and r < p+1. We note that also here, like in the dynamic setting, the value r = p+1 is a special value that plays a significant role.

Deterministic methods vs. probabilistic ones. In the dynamic setting we concentrated on deterministic algorithms. In the static setting, however, not having the freedom of deciding on the allocation of variants in the next segment based on the findings in the previous segments, we restrict our discussion to probabilistic algorithms. This is because we have been unable to find a static deterministic algorithm that is more efficient than the exponential scheme in §3.2. One obvious open problem is proving that subexponential deterministic static schemes are impossible, or the converse.

Note that by relaxing the deterministic incrimination demand to a probabilistic one, we are able to break the barrier of r = p + 1 and achieve such a relaxed incrimination even with the minimal size of marking alphabet, r = 2, though, as shown later, with a very high price in m.

4.2 Lower Bounds

Under the assumptions and definitions given in §4.1, the following holds:

Lemma 1. There exists a constant b > 1, depending solely on p and r, such that for any $t \in T$, $M(C(t), C^*) \xrightarrow[m \to \infty]{b} r$ a.e.

Proof. For every $u \in U$, let $x(u)_j$, $1 \le j \le m$, be the following indicator random variables:

$$x(u)_j = \begin{cases} 1 & \text{if } C(u)_j = C_j^* \\ 0 & \text{otherwise} \end{cases}$$
 (1)

We also define ξ_p^r to be the random variable that counts the number of different letters in a word of length p over an alphabet of size r and we let

$$\alpha_{p,r}(k) = \Pr[\xi_p^r = k] \ . \tag{2}$$

With this, we get that for every $t \in T$,

$$E[x(t)_j] = Pr[x(t)_j = 1] =$$
(3)

$$\sum_{k=1}^{r} Pr\left[C(t)_{j} = C_{j}^{*} \mid |\{C(t)_{j} : t \in T\}| = k\right] \cdot \alpha_{p,r}(k) = \sum_{k=1}^{r} \frac{\alpha_{p,r}(k)}{k} . \tag{4}$$

Since $\alpha_{p,r}(k) > 0$ for all $1 \le k \le r$, it follows that

$$E[x(t)_j] = \frac{b}{r} \tag{5}$$

for some b>1 that depends solely on p and r. Finally, since $x(t)_j$ are independent and have the same distribution, and $M(C(t), C^*) = \frac{1}{m} \cdot \sum_{j=1}^m x(t)_j$, then, by The Strong Law of Large Numbers, it converges to its expected value a.e.

Theorem 3. There exists m sufficiently large for which

$$Pr\left[M(C(u), C^*) < \max_{t \in T} M(C(t), C^*) \quad \forall u \in U \setminus T\right] > 1 - \varepsilon. \tag{6}$$

Proof. In view of Lemma 1, we may approximate the maximal score of the traitors by

$$\mu := \frac{b}{r} \ . \tag{7}$$

Hence, we aim at showing that, for sufficiently large m,

$$Pr[M(C(u), C^*) < \mu \quad \forall u \in U \setminus T] > 1 - \varepsilon.$$
(8)

Fixing $u \in U \setminus T$ and denoting $a = Pr[M(C(u), C^*) \ge \mu]$, inequality (8) is equivalent to $(1-a)^{n-p} > 1-\varepsilon$, and this inequality holds if

$$a < \frac{\varepsilon}{n}$$
 (9)

Hence, we aim at finding m for which (9) holds. To this end, we observe that

$$a = Pr \left[\frac{1}{m} \sum_{j=1}^{m} x(u)_j \ge \mu \right]$$
 (10)

where $x(u)_j$ were defined in (1). Since $x(u)_j \sim B(\frac{1}{r})$, $1 \leq j \leq m$, and are independent, we may apply the normal approximation and conclude that

$$X := \frac{r \cdot \sum_{j=1}^{m} x(u)_j - m}{\sqrt{m(r-1)}} \sim N(0,1)$$
 (11)

for sufficiently large m. Hence, in view of (7), (10) and (11), the inequality in (9) takes the form $Pr\left[X \geq \frac{b-1}{\sqrt{r-1}}\sqrt{m}\right] < \frac{\varepsilon}{n}$, and that holds if

$$m > M_1 := \frac{r-1}{(b-1)^2} \cdot \left(\operatorname{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2$$
 where $\operatorname{erfc}(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} dt$. (12)

The proof of Theorem 3 is constructive, in the sense that it provides an expression for the lower bound M_1 . That lower bound involves the constant b which may not be computed explicitly. However, by (4) and (5),

$$b = r \cdot \sum_{k=1}^{r} \frac{\alpha_{p,r}(k)}{k} , \qquad (13)$$

and that may be easily evaluated using the following self-explanatory formulae for computing $\alpha_{p,r}(k)$, (2):

$$\alpha_{1,r}(k) = \begin{cases} 1 & \text{if } k = 1 \\ 0 & \text{if } k \neq 1 \end{cases} ; \quad \alpha_{p,r}(0) = 0 ;$$
 (14)

$$\alpha_{p+1,r}(k) = \frac{k}{r} \cdot \alpha_{p,r}(k) + \left(1 - \frac{k-1}{r}\right) \cdot \alpha_{p,r}(k-1) , \qquad (15)$$

where $p, r \ge 1$ and $1 \le k \le r$.

The case $r \ge p+1$. It is always safe to say that $\max_{t \in T} M(C(t), C^*) \ge \frac{1}{p}$. Hence, in this case we may take in (7) $b = \frac{r}{p}$ instead of (13). Substituting this explicit value for b in (12), we arrive at the following lower bound for m:

$$m > M_2 := \frac{(r-1)p^2}{(r-p)^2} \cdot \left(\operatorname{erfc}^{-1}\left(\frac{\varepsilon}{n}\right)\right)^2 . \tag{16}$$

Clearly, the lower bound in (16), M_2 , is worse than that in (12), M_1 . Let us consider the asymptotic behavior of M_2 for large values of p and see how it compares to that of M_1 . It turns out that the answer to these questions depends on the relation between r and p:

• If $r = \gamma \cdot p$ where $\gamma = \text{Const} > 1$, then, by (16),

$$M_2 = \frac{\gamma \cdot p - 1}{(\gamma - 1)^2} \cdot \left(\operatorname{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \sim \operatorname{Const} \cdot p \cdot \left(\operatorname{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2.$$

The lower bound that was obtained in [2] under the assumption r=4p was $M_{\rm CFN}=\frac{4}{3}p\log_2\left(\frac{n}{\varepsilon}\right)$. Comparing the three bounds in the case r=4p, we find that $M_1 < M_2 < M_{\rm CFN}$ while $\frac{M_2}{M_1} \approx 1.377$ and $\frac{M_{\rm CFN}}{M_1} \approx 3.629$. Therefore, in practice, M_1 should be used.

• If $r = p + \gamma$ where $\gamma = \text{Const}$, then, by (16),

$$M_2 = \frac{(p+\gamma-1)\cdot p^2}{\gamma^2} \cdot \left(\mathrm{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \sim \mathrm{Const} \cdot p^3 \cdot \left(\mathrm{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \ .$$

However, as numerical computations indicate, M_2 becomes meaningless in this case and M_1 should be evaluated in order to get a reasonable estimate for the required lower bound on m.

The case $r \leq p$. The majority scheme works also when $r \leq p$, even with the minimal r, i.e., r = 2. However, the lower bound M_1 increases dramatically and the scheme becomes impractical; for example, when r = 2, $M_1 = 4^{p-1} \cdot \left(\operatorname{erfc}^{-1}\left(\frac{\varepsilon}{n}\right)\right)^2$. The reason for that is that the smaller r is, the better are the chances that the traitors would have in each segment a full or almost full set of variants. In other words, the smaller r is, the closer C(T) is to Σ^m and then the traitors could produce almost any version. For instance, if $p > r \ln r$ then, by the coupon collector problem, there are good chances that in a given segment j, the traitors have the complete set of variants, $\{C(t)_j: t \in T\} = \Sigma$. However, those segments in which the pirate lacks some variants, enable the scheme to relate a given black market copy, C^* , to T, provided that there are enough of those segments, i.e., provided that m is sufficiently large. Hence, in applications where the size of of the watermarking alphabet is strictly restricted, more effective methods should be applied.

5 Open Problems

It is important to understand the underlying performance considerations which one needs to consider: bandwidth, storage and computation time. Some of the published results on various broadcast problems are seemingly irrelevant because they do not deal with the performance characteristics of the solution. One important task is to give a unified analysis of the various solutions proposed in the literature.

As for the present study, the open problems that it raises are as follows:

- Devising a probabilistic algorithm in the dynamic model. There are two settings to consider in this context: (a) known allocation of codewords (the pirate knows the codewords of all users and not just of those he controls), and (b) oblivious allocation of codewords.
- 2. Finding a deterministic dynamic algorithm based on a minimal alphabet, r=p+1, with a convergence time that is polynomial in p.
- 3. Proving or disproving that any deterministic static scheme is exponential (in the number of segments m).

Acknowledgments

The authors gratefully acknowledge interesting and valuable conversations with Omer Berkman, Jacob Goldberger and Benny Pinkas.

References

- D. Boneh and J. Shaw, Collusion-Secure Fingerprinting for Digital Data, IEEE Transactions on Information Theory, Vol. 44, 5, 1998, pp. 1897–1905 (see also Proc. Crypto 95, Springer LNCS 963, 1995, pp. 452-465).
- B. Chor, A. Fiat, and M. Naor, Tracing Traitors, Proc. Crypto 94, Springer LNCS 839, 1994, pp. 257–270.
- I.J. Cox, J. Kilian, T. Leighton and T. Shamoon, A Secure, Robust Watermark for Multimedia, Information Hiding, Springer LNCS 174, 1996, pp. 185–226.
- C. Dwork, J. Lotspiech and M. Naor, Digital signets: Self-Enforcing Protection of Digital Information, 28th Symposium on the Theory of Computation, 1996, pp. 489-498.
- A. Fiat and M. Naor, Broadcast Encryption, Proc. Crypto 93, Springer LNCS 773, 1993, pp. 480–491.
- Paul Kocher, Cryptography Research, http://www.cryptography.com/dpa/ index.html
- M. Naor and B. Pinkas, Threshold Traitor Tracing, Proc. Crypto 98, Springer LNCS 1462, 1998, pp. 502-517.
- 8. B. Pfitzmann, *Trials of Traced Traitors*, Information Hiding, Springer LNCS 1174, 1996, pp. 49–64.
- D. R. Stinson and R. Wei, Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes, SIAM Journal on Discrete Mathematics Vol. 11, 1, 1998, pp. 41-53.
- 10. J. Schwenk, J. Ueberberg, Tracing Traitors using Finite Geometries, manuscript.

Efficient Methods for Integrating Traceability and Broadcast Encryption

Eli Gafni^{1*}, Jessica Staddon^{2*}, and Yiqun Lisa Yin^{3*}

University of California at Los Angeles eli@cs.ucla.edu
Bell Laboratories Research Silicon Valley jstaddon@yahoo.com
NTT Multimedia Communications Laboratories yiqun@nttmcl.com

Abstract. In many applications for content distribution, broadcast channels are used to transmit information from a distribution center to a large set of users. Broadcast encryption schemes enable the center to prevent certain users from recovering the information that is broadcast in encrypted form, while traceability schemes enable the center to trace users who collude to produce pirate decoders. In this paper, we study general methods for integrating traceability and broadcasting capability. In particular, we present a method for adding any desired level of broadcasting capability to any traceability scheme and a method for adding any desired level of traceability to any broadcast encryption scheme. To support our general methods, we also present new constructions of broadcast encryption schemes which are close to optimal in terms of the total number keys required. Our new schemes are the first to be both maximally resilient and fully scalable.

1 Introduction

In many applications for content distribution, broadcast channels are used to transmit a message from a distribution center to a large set of users. It is often desirable for the center to be able to exclude certain users from recovering the message that is broadcast in encrypted form. One such example is the pay television industry, in which only privileged users (i.e., active subscribers) are permitted to view shows. Many solutions to this problem have been proposed using broadcast encryption schemes [3,9,10,4,5,15,13,20,11,19,16,6,1]. In such schemes, keys are allocated to users in such a way that broadcasts can be made to selected sets with security. To broadcast to the selected set, a subset of the encryption keys is used to encrypt the message based on the protocol being used. The basic attribute of a broadcast encryption scheme is its broadcasting capability, which is generally measured by the number of users that can be prevented from recovering the message from the broadcast.

^{*} Most of this work was done while the authors were working at RSA Laboratories.

Clearly, security is an important attribute of broadcast encryption schemes. Two commonly used measures of security are resiliency [9] and traceability [7,17,12]. A scheme is said to have m-resiliency if no set of at most m excluded users can pool their keys together to recover a message from a broadcast. A scheme is said to have c-traceability if when a set of at most c users (who are not necessarily excluded) pool their keys together to construct a "pirate decoder", at least one of the users involved can be identified by examining the keys in the decoder. Traceability can offer protection against the piracy that is often a serious problem in content distribution applications.

Although a natural goal for constructing broadcast encryption schemes is to have both high broadcasting capability and high traceability, these two attributes have been studied separately in the past, with the exceptions of [13,18]. In [13], Staddon determines the traceability of various specific broadcast encryption schemes and proves lower bounds on the traceability of certain (protocol dependent) broadcast encryption schemes. In [18], Stinson and Wei develop the first method for combining the two attributes by adding broadcasting capability to a given traceability scheme. Their method is quite general in that the construction is based on an arbitrary traceability scheme.

The first contribution of this paper is to study general methods for integrating traceability and broadcasting capability. In particular, we approach the integration problem from both directions: (1) we develop the first method for adding any desired level of traceability to an arbitrary broadcast encryption scheme; (2) we develop a new method for adding any desired level of broadcasting capability to an arbitrary traceability scheme.

The central idea behind our method for adding traceability to broadcast encryption schemes is that using "randomness" when allocating keys to users allows the users' key sets to be dispersed, and hence, is conducive to traceability. Based on this observation, our method adds a "dimension" of randomness to an underlying broadcasting encryption scheme. In the other direction, our method adds adding broadcasting capability to a traceability scheme. The main idea behind the latter method is to leverage on the inherent broadcasting capability in the underlying traceability scheme. We show that by exploiting such inherent broadcasting "structure", significant efficiency improvements can be achieved over the method in [18].

For both of the general methods that we present here, keys are allocated to users according to a certain matrix. The keys appear to be randomly assigned to users along one dimension of the matrix but well structured along the other dimension. The random dimension contributes to traceability and the structured dimension contributes to broadcasting capability. Hence, the two methods are complementary to each other and are conceptually quite simple.

An important feature of these methods is their preservation of the properties of the underlying broadcast encryption schemes. In addition to resiliency, another important property is *full scalability*. This means that the set of keys for each existing user remains unchanged when new users are introduced into the system. While scalability is clearly a desirable attribute for any large con-

tent distribution system, it has been largely ignored in the context of broadcast encryption. Our second contribution is to propose two new maximally-resilient fully-scalable broadcast encryption schemes, to which one may add traceability by our general method.

One of our schemes is based on a geometric construction and the other on an algebraic construction. Both schemes employ the so-called "OR protocols" [1,10,11], which have the desirable property of yielding maximally resilient schemes. We show that our new schemes are close to optimal with respect to the total number of keys by proving a lower bound that almost matches this number. This lower bound is obtained by demonstrating a concise combinatorial characterization of broadcast encryption systems with OR protocols. These results establish a relationship between the number of keys per user, r, and the total number of keys, K. Although individual bounds on r and K have been shown [11], the relationship between the two has not been studied prior to our work.

The organization of this paper is as follows: Section 2 contains notation and definitions. Section 3 summarizes related work in broadcast encryption and traceability schemes. Section 4 describes the new broadcast encryption schemes and proves a tight lower bound relating the number of keys per user and the total number of keys. Section 5 presents the general methods for integrating broadcasting capability and traceability.

2 Preliminaries

In this section, we provide the notation and definitions for broadcast encryption schemes and their attributes. At a very high level, a broadcast encryption scheme consists of users, keys, a key allocation method for assigning keys to users, and a broadcast protocol that the center uses to transmit information to certain sets of users.

Let $\{u_1, ..., u_n\}$ denote the set of all users. We call the users who have the permission to receive a message that's broadcast in encrypted form, the set of privileged users, and the users who don't have permission, excluded users. We use \mathcal{P} to denote the collection of privileged sets of users and m to denote the number of excluded users. So, \mathcal{P} is the collection of all subsets of users of size n-m.

Let $S = \{k_1, ..., k_K\}$ denote the set of all keys. The set of keys assigned to user u, is denoted by $U \subseteq S$. Since we mostly focus on the maximum number of keys per user as an important measure of the efficiency, it is without loss of generality that we assume all users have the maximum number of keys, r. That is, for each user u, |U| = r.

For a set of privileged users, $P \in \mathcal{P}$, the set of keys that the center uses to broadcast to P will be denoted by $S_P \subseteq S$. The number of transmissions for a broadcast encryption scheme is defined to be $t = \max_{P \in \mathcal{P}} |S_P|$. This is the number of keys used in the communication.

In most applications for content distribution, the center first establishes a broadcast key, B_P , with the set of privileged users P, and encrypts subsequent broadcasts with the broadcast key. For each privileged set P, there is a broadcast protocol which defines which subsets of keys in S_P are used to encrypt and recover B_P . Hence, a protocol yields an access structure on S_P because it defines which subsets of S_P suffice to recover B_P . Therefore, to implement any protocol for broadcasting to P, one can use the keys in $S_P = \{k_1, ..., k_t\}$, to generate shares, $B_P^1, ..., B_P^t$, according to the access structure and the choice of secret sharing scheme (see [14] for more on access structures and secret sharing). Each share B_P^i is then encrypted in a computationally secure way, so that key k_i is necessary to decrypt it. We assume that a user u for which $k_i \notin U$ gains no information about B_P^i from its encrypted form.

In this paper, we concentrate on OR protocols. If the center is broadcasting to a set P with an OR protocol, then a user needs only one out of the t keys in S_P to decrypt B_P . Consequently, to implement OR protocols, one can use a (1,t)-threshold scheme to generate the shares¹. We focus on OR protocols because a broadcast encryption scheme that employs them is secure against arbitrary coalitions of excluded users. An excluded user has none of the keys in S_P , therefore a coalition of excluded users of arbitrary size still cannot recover B_P (consequently, OR protocols are said to be arbitrarily resilient).

Clearly, many other protocols are possible. In fact, any formula of a certain form (see [11]) defines a protocol. For example, in an AND protocol, all the keys in S_P are necessary to recover B_P . An AND protocol can be implemented with a (t,t)-threshold scheme.

As discussed earlier, traceability offers a form of security that is complementary to resiliency. Traceability protects against a coalition of users, \mathcal{C} , who build a "pirate decoder", F. The decoder can be modeled as a subset of their pooled keys. That is, $F \subseteq \bigcup_{u \in \mathcal{C}} U$, such that $|F| \geq r$. In addition to the basic components in broadcast encryption, a traceability scheme also consists of an algorithm which identifies one user in \mathcal{C} by analyzing the keys in F. Informally, we say that a scheme has c-traceability if when the size of \mathcal{C} is at most c, at least one of the users involved in coalition can be identified with very high probability. In other words, an innocent user will be identified as "guilty" with only negligible probability. This is called a c-resilient traceability scheme in [7].

Definition 1. Let C be any coalition of at most c users who produce a pirate decoder F. A scheme is called a c-traceability scheme if for any user u, such that for all users $w \neq u$ the following inequality holds:

$$|U\cap F|\geq |W\cap F|$$

then the probability that u is not a member of the coalition C is negligible.

Another desirable attribute of a broadcast encryption or a traceability scheme is that it scale well. This means that as the number of users grows, only a small

¹ Note that we've described a natural way to implement an *OR* protocol and from the results in [10] it follows that it is as efficient as possible.

amount of rekeying is necessary for the old users. In certain systems, it might be required that no rekeying is needed.

Definition 2. A scheme is fully scalable or has full scalability if when new users are added, no rekeying of existing users is necessary.

For ease of notation, we assume that when discussing a scheme, the parameters of the scheme (such as the number of keys per user, etc.) are represented by the notation summarized in Table 1, unless otherwise specified.

Summary of Terms and Notation

- $\{u_1, ..., u_n\}$ is the set of all users.
- $\{k_1, ..., k_K\}$ is the set of all keys.
- $-S_P$ is the set of keys used to broadcast to privileged set P.
- $-B_P$ is the message (e.g., a broadcast key) that is broadcast to P in encrypted form.
- -n is the total number of users.
- -K is the total number keys.
- -r the number of keys per user.
- -m is the number of users who are excluded.
- t is the number of transmissions. Note that $|S_P| \leq t$.
- -c is the traceability of the scheme.
- OR Protocol for Broadcasting to P: Any one of the keys in S_P suffices to recover B_P from the broadcast.
- AND Protocol for Broadcasting to P: All of the keys in S_P are necessary to recover B_P from the broadcast.

Table 1. Summary of Terms and Notation

3 Related Work

3.1 Broadcast Encryption

The early works in broadcast encryption are [3,9,10,4]. In [3], a one-time broadcast encryption scheme is presented. It can be used once with security as information about each user's key is leaked to the privileged set during broadcast. Our model for broadcast encryption is a formalization of the one in [9]. In [9], the concept of resiliency is formalized, and broadcast encryption schemes of various resiliencies are constructed. In [10], the authors consider broadcast encryption schemes with OR protocols (although this terminology is not used) and prove that the entropy of a broadcast is at least the size of the entropy of the message times the number of users in the privileged set. The work in [4] concentrates on broadcast encryption schemes in which only one transmission is needed by

the broadcasting center (called *zero-message schemes*) and on schemes in which users interact. Some information theoretic lower bounds are also derived.

In several subsequent works [5,15,13,11,19,16], the trade-off between communication cost and storage in broadcast encryption is studied. Many new schemes are proposed, some of which are combinatorial in nature. In [5,15,19,16], the trade-off is measured using an information theoretic ratio, while in [13,11] it is measured through a comparison of the number of keys (per user and in total) versus the number of keys used in the communication. Consequently, schemes that are optimal under one measurement may not be optimal under the other. We note that most of the schemes in [5,13,19,16] use (as a component) a construction in [9] that does not scale very well.

The recent work in [1] focuses on constructing broadcast encryption systems in which the user storage is very limited. In their proposed systems, the reduction in storage is achieved by allowing a controlled number of excluded users to receive the broadcast. They prove some lower bounds under this framework and and present an algorithm for efficiently finding such schemes while minimizing the communication cost.

A quite different approach to solving the problem of broadcast encryption appears in [20,6]. The model differs from all of the above mentioned works in that when some user is removed from the system, keys of existing users are updated (called rekeying). In the Internet draft [20], a hierarchical tree-based scheme is recommended for use in a broadcast encryption system. The system is maximally resilient but not fully scalable. This work is later built upon in [6], which demonstrates a method for reducing center's storage in the tree-based scheme by considering the trade-off between storage and the rekeying communication cost.

Our model is consistent with those in [13,11,1], and is a formalization of the one in [9]. These works ([13,11,1]) focus on two important quantities: the number of keys per user, and the total number of keys. These are important quantities because they give a concrete bound on storage requirements which is very useful for implementation. In addition, when OR protocols are used (as in this paper), the resulting broadcast size is just a multiple of the number of transmissions, therefore bandwidth is a straightforward calculation. Although the schemes in this paper are not tight with the bound in [11], no broadcast encryption schemes with OR protocols are known that are tight with this bound for the number of transmissions required by these schemes (see Table 1). In addition, there is evidence that the bound in [11] is not tight for certain values of the parameters. For example, if t is on the order of \sqrt{n} and m is small, then to be tight with the bound means that K and r must be essentially 1, which is clearly impossible. Further, we emphasize that [11] does not establish a relationship between r and K, but rather, it proves that both r and K are $\Omega(\binom{n}{m})^{1/t}$. In addition, our schemes are fully scalable and allow the implementor complete control over the number of keys per user. These are not features of any other maximally resilient broadcast encryption scheme.

In overall comparison with the previous work, we emphasize that our new broadcast encryption schemes (in Section 4) are the first that are both maximally resilient and fully scalable under this model. In addition, the schemes are very flexible in terms of the number of keys per user. We note that in this paper we do not consider other models such as one-time schemes [3], zero-message schemes [4], and schemes that allow rekeying [20,6].

3.2 Traceability Schemes

Traceability schemes are first introduced in [7] and further studied in [17]. Several constructions for traceability schemes are given and lower bounds on the number of keys per user and the total number of keys are proven.

A generalization of traceability called threshold traceability, is considered in [12]. Threshold traceability schemes are designed to trace the source of a pirate decoder which can decrypt with only a probability larger than some threshold. By relaxing the decryption probability requirement, a significant reduction in storage and communication is achieved.

Our model for traceability schemes is the same as the one in [7,17]. The methods for integrating broadcast encryption and traceability (in Section 5) can be extended to the generalized model in [12].

3.3 Integrating Broadcast Encryption and Traceability

There are only two previous works [13,18] that study the integration of broadcast encryption and traceability. In [13] the traceability of various specific broadcast encryption schemes is determined and lower bounds on the traceability of certain (protocol dependent) broadcast encryption schemes is proven. The focus of this work is to determine the traceability of certain broadcast encryption schemes, rather than to demonstrate how to achieve a certain level of traceability with a specific broadcast encryption scheme.

In [18] the model of traceability is a generalization of the model in [7]. They allow decoders to hold any number of keys and they allow the set of excluded users to be a *proper* subset of the complement of the set of privileged users. In [18], broadcasting capability is added to a traceability scheme by using a construction in [9] to expand each key into a set of keys. Our method (Method 2 in Section 5) differs from their method in that we take full advantage of the *inherent* broadcasting capability in the underlying traceability scheme, and therefore our method requires much less keys per user than [18] in most situations. We remark, however, that the model in [18] is more general, and hence their method may be applicable to more situations than our method which follows the earlier model in [7].

4 Optimal Broadcast Encryption Schemes with OR Protocols

In this section we describe two new constructions for broadcast encryption schemes with OR protocols. Recall that OR protocols are desirable because of their inherent resiliency. The first construction, which we call the *cube scheme*, is based on a geometric construction. The second one, which we call the *polynomial scheme*, is based on an algebraic construction. Both schemes are fully scalable and m-resilient (due to the use of OR protocols. In particular, we emphasize that when new users are added, the set of keys for any existing user remains unchanged. We also show that both schemes are close to optimal in terms of the total number of keys by proving a matching lower bound.

4.1 The Cube Scheme

The cube scheme is a parameterized scheme. For a fixed number of keys per user, r, the construction is based on an r-dimensional cube. Informally speaking, users are represented by *entries* of the cube (i.e., points), and keys are represented by *slices* of the cube (i.e., subspaces of dimension r-1).

First we describe the case in which r=2, as it is easier to understand and suggests a natural generalization. Consider a $n^{1/2} \times n^{1/2}$ square, and associate each of the n users with an entry in this square indexed by (i_1, i_2) , where $i_1, i_2 \in \{1, 2, ... n^{1/2}\}$. For $1 \le i \le n^{1/2}$, let C_i denote the set of users in column i and let R_i denote the set of users in row i. For each i, we create two unique keys and allocate one of the keys only to the users in C_i and allocate the other only to the users in R_i . Therefore, each user has exactly 2 keys. To exclude a given user u, the center broadcasts according to an OR protocol with all the keys except the 2 keys stored by user u. Since each two users share at most 1 key, every user except u can receive the broadcast.

We can easily generalize the above scheme to dimension r by associating each user with an entry in an r-dimensional cube. An r-dimensional cube has entries indexed by r-tuples, $(i_1, ..., i_r)$ where each $i_j \in \{1, 2, ..., n^{1/r}\}$. We define a slice of the cube to be the (r-1)-dimensional analog of rows and columns, that is, a subspace of dimension r-1. More precisely, for each pair (j, w) such that $1 \le j \le r$ and $1 \le w \le n^{1/r}$, we define a slice, $S_{j,w}$:

$$S_{j,w} = \{(i_1, i_2, ..., i_r) : i_j = w\}.$$

In other words, a slice consists of all the r-tuples which are identical in the jth entry. As in the 2-dimensional case, we create a unique key for each slice. Therefore, each user has exactly r keys. To exclude a given user u, the center broadcasts according to an OR protocol with all the keys except the r keys that u has. Since each pair of users share at most r-1 keys, every user except u can recover B_P from the broadcast. Note that the cube scheme can exclude one user.

We now present a simple extension of the above construction to exclude m users by making "copies" of the cube scheme. Specifically, we assign independent keys to m different r-dimensional cube schemes, therefore each user has rm keys in total. We can exclude m users, $\{u_1, u_2, ..., u_m\}$ by excluding the r keys that user i has in the ith cube scheme. The broadcast protocol is then an AND on the union of the sets of keys left in each cube scheme. The resulting scheme is still 1-resilient. In summary, for this scheme, the total number of keys is $K = mrn^{1/r}$, the number of keys per user is mr, the number of transmissions is K - mr, and the resiliency is 1.

Finally, we note that the cube scheme and it extension scale well as the number of users grows. For example, we can add $n^{(r-1)/r}$ users by expanding the cube by the size of one slice. This requires the addition of only one new key. The new users are given that new key and old keys corresponding to the other slices in which they are contained. No rekeying is necessary for old users. This is significantly better than in the previously known schemes. For example, in the OR scheme in [11], there is a key for each set of $\frac{n-m}{t}$ users. Therefore, adding one new user necessitates the creation of $\binom{n-1}{n-m}$ new keys, and each old user needs $\binom{n-2}{n-m}$ new keys.

4.2 The Polynomial Scheme

The polynomial scheme described in this section is a parameterized scheme depending on both r, the number of keys per user, and m, the number of excluded users. The scheme uses a set system construction² based on polynomials over a finite field. Speaking informally, users are represented by polynomials and keys are represented by points on the polynomials.

Let p be a prime larger than r, and let A be a subset of the finite field F_p of size r. Consider the set of all polynomials over F_p of degree at most $\frac{r-1}{m}$. (For simplicity, we assume that m|(r-1).) There are $p^{\frac{r-1}{m}+1}$ such polynomials. We associate each of the n users with a different polynomial. Therefore, p needs to satisfy the condition that $p^{\frac{r-1}{m}+1} \geq n$, or equivalently, $p \geq n^{\frac{r}{m-1+m}}$. The keys are created and assigned to users as follows: We create a unique key, $k_{(x,y)}$, for each pair (x,y) where $x \in A$ and $y \in F_p$. Note that the polynomials may be public information, as knowledge of a user's polynomial reveals only the *indices* of that user's keys, not the keys themselves. For a user u who is associated with a given polynomial f, u is allocated all the keys in the set $\{k_{(x,f(x))}|x \in A\}$. Since any two of the polynomials intersect in at most $\frac{r-1}{m}$ points, it follows that any two users share at most $\frac{r-1}{m}$ keys. This ensures that if all the keys belonging to the m excluded users are removed, then each privileged user will still have at least 1 key. Therefore, the center can broadcast with an OR protocol to any set of n-m users. In summary, the total number of keys is $K=rp \geq rn^{\frac{m}{r-1+m}}$, the number of keys per user is r, the number of transmissions is at most K-r, and the resiliency is m.

² The construction appeared in [2] in a purely combinatorial context.

This scheme is also fully scalable, since increasing the size of the field, F_p , allows significantly more users to be added with no rekeying of the old users. For example, if K is doubled, then $2^{\frac{r-1}{m}+1}$ more users can be added to the scheme. The new users will get some of the new keys and some of the old, while the old users key sets will remain unchanged.

Finally, we note that for certain values of the parameters this scheme may be closely related to the cube scheme of the previous section.

4.3 Lower Bound on the Total Number of Keys

In this section, we establish a lower bound on the total number of keys in a broadcast encryption scheme in terms of the number of keys assigned to each user. This lower bound shows that the total number of keys is close to optimal in both the cube scheme and the polynomial scheme. To prove the bound, we first demonstrate a combinatorial characterization of broadcast encryption schemes with OR protocols.

Lemma 3. A collection of n sets can be used as a broadcast encryption scheme with OR protocols that can exclude any set of m users if and only if

$$\forall U_{i_1}, ..., U_{i_{m+1}} \text{ distinct, } U_{i_1} \not\subseteq \bigcup_{j=2}^{m+1} U_{i_j}$$

Proof: \Rightarrow : Assume we have such a broadcast encryption scheme and there exists a set of m+1 users, $u_1,...,u_{m+1}$, such that $U_1 \subseteq \bigcup_{j=2}^{m+1} U_j$. Then, if OR protocols are used, at least one of $u_2,...,u_{m+1}$ will be able to recover the message from a broadcast to u_1 . This is a contradiction.

 \Leftarrow : If for every set of m users $u_1, ..., u_m$ and for every user, u, outside of this set, $U \nsubseteq \bigcup_{j=1}^m U_j$, then to broadcast to $P = \{u_{m+1}, ..., u_n\}$, let $S_P = S - \bigcup_{i=m+1}^n U_i$. This S_P (or possibly even a subset of it) can be used to broadcast to P with OR protocols. \square

The following result by Erdös, Frankl, Füredi [8] is very useful in determining the relationship between the parameters of a set system satisfying the condition in the previous lemma.

Theorem 4 ([8]). Let $U = \{k_1, k_2, ..., k_K\}$ be a set of K elements. Let $U_1, ..., U_n$ be a collection of n subsets of U such that $\forall j, |U_j| = r$, and $\forall U_{i_1}, ..., U_{i_{m+1}}$ distinct, $U_{i_1} \nsubseteq \bigcup_{j=2}^{m+1} U_{i_j}$, then

$$n \le \frac{\binom{K}{\lceil r/m \rceil}}{\binom{r-1}{\lceil r/m \rceil - 1}}$$

Combining Lemma 3 and Theorem 4, we can establish a relationship between the total number of keys and the number of keys per user.

Theorem 5. In a broadcast encryption scheme with OR protocols, the total number of keys, K, is $\Omega((n/m)^{m/r}r)$, where $r \geq m$ is the number of keys per user and m is the number of users that can be excluded in the scheme.

Proof: From Lemma 3, it follows that any broadcast encryption scheme with OR protocols must satisfy the condition of Theorem 4. Then the lower bound on K can be easily derived from the inequality given in Theorem 4. \square

The lower bound given in Theorem 5 enables one to first choose the number of keys per user when constructing a broadcast encryption scheme (e.g. based on the storage capabilities of a smart card), and then determine the minimum total number of keys that is necessary. Indeed, this is the approach that we have used in both the cube scheme and the polynomial scheme. Table 2 summarizes these schemes. Based on our lower bound, it is easy to see that both schemes are close to optimal in terms of the total number of keys. We remark that fixing the number of keys per user ahead of time (so that it is independent of the total number of users) is very useful in constructing fully scalable broadcast encryption schemes.

Scheme	Number of users can exclude	Resiliency	number	Number of keys per user
r-dimensional cube scheme	1	1	$rn^{1/r}$	r
m copies of the cube scheme	m	1	$mrn^{1/r}$	mr
(r, m)-polynomial scheme	m	m	$\geq n^{\frac{m}{r-1+m}}r$	r

Table 2. A Summary of the Broadcast Encryption Schemes in Sections 4.1 and 4.2.

We also note that for certain values of the parameters, Theorem 5 may yield a larger bound on K than is proven in [11], and is, therefore, an improvement. For example, when t is large, the bound in [11] $(K \text{ is } \Omega(\binom{n}{m})^{1/t})$ is only trivially true, as it is quite small.

5 Integrating Traceability and Broadcast Encryption

In this section, we present two methods for integrating traceability with broadcasting capability. Our methods are both efficient and conceptually quite simple.

In Section 5.1 we describe a method that adds any desired level of traceability to any given broadcast encryption scheme, \mathcal{B} . A scheme constructed by this method can be viewed as a two dimensional matrix, in which broadcasting capability is drawn from one dimension, and traceability from the other. In other words, if we assume that all the keys in \mathcal{B} are arranged in one column, then the method extends the column of keys into a matrix in such a way that the horizontal dimension contributes traceability.

In Section 5.2 we describe a method that adds any desired level of broadcasting capability to any traceability scheme, \mathcal{T} . A scheme constructed with this method can also be viewed as a two dimensional matrix. If we arrange all the keys in \mathcal{T} in one row, then this method extends this row of keys into a matrix in such a way that the vertical dimension contributes broadcasting capability.

Together, these complementary approaches solve the problem of integrating traceability and broadcasting capability from both directions.

5.1 Adding Traceability to Broadcast Encryption Schemes

We first consider how much traceability is inherent in a broadcast encryption scheme.

Lemma 6. Any broadcast encryption scheme that can exclude $m \ (m \ge 1)$ users has at least 1-traceability. In addition, a broadcast encryption scheme that can exclude m users may have no more than 1-traceability.

Proof: The first statement follows from the definitions. To prove the second statement, it suffices to produce a broadcast encryption scheme with 1-trace-ability. In [13], a scheme using AND protocols is described and it's proven that the scheme has 1-traceability for sufficiently large n. \square

From this lemma, it is clear that the traceability of an arbitrary broadcast encryption scheme can be quite limited. We now turn to our method for adding traceability to an arbitrary broadcast encryption scheme.

The schemes in [7] gain traceability from "randomness" in the key assignments. The random nature of the key assignments forces the key sets of the individual users to be distinct enough that traitors can be identified with high probability upon examination of the keys in a decoder. In most broadcast encryption schemes, however, keys are assigned to users in a very structured way. Therefore, the central idea in our method is to incorporate some randomness into the way in which the keys are assigned to users in a broadcast encryption scheme. Our method is motivated by the constructions of traceability schemes in [9]. The method is described in Table 3.

The following theorem gives the precise parameter values for an implementation of our method using the "open one-level" scheme of Fiat and Naor [9], which defines a practical way of assigning the keys in step 2 of Method 1 using hash functions.

Theorem 7. Let \mathcal{B} be a broadcast encryption scheme with parameters (n, m, K, r, t). If $r > 4c^2 \log n$, then there exists a broadcast encryption scheme, \mathcal{B}' , which has c-traceability and parameters (n, m, K', r', t'), where $K' = 2c^2K$, r' = r, and $t' = 2c^2t$.

Proof: All the assertions about \mathcal{B}' except its c-traceability follow from the construction of Method 1 given in Table 3. The argument for traceability is very similar to the argument for the "open one-level" scheme in [7]. In particular, if we set $h = 2c^2$ and a pirate decoder contains at least $s > 4c^2 \log n$ keys, then the probability that a user who has at least $\frac{s}{c}$ keys in common with the decoder is innocent, is negligible. By definition, \mathcal{B}' has c-traceability. \square

Method 1

Input:

- a broadcast encryption scheme, B
- an integer, c, the desired level of traceability

Output:

- a broadcast encryption scheme, \mathcal{B}' , with c-traceability

Construction:

- 1. Let $\{k_1, ..., k_K\}$ be the set of keys in scheme \mathcal{B} . For each key k_j , create a set of h keys $W_j = \{k_{j,1}, k_{j,2}, ..., k_{j,h}\}$, where h (value to be determined) depends on c.
- 2. If a user u has key k_j in \mathcal{B} , then in \mathcal{B}' , u gets one key randomly chosen from the set W_j .
- 3. To broadcast a secret B_P to a set of privileged users, P, where $S_P = \{k_{i_1}, ..., k_{i_t}\}$, the center first generates shares $B_P^1, ..., B_P^t$, according to the protocol used for P in \mathcal{B} (as described in Section 2). Then for each key k_{i_j} in S_P , the center encrypts B_P^j with each of the keys in the set W_j .

Table 3. A method for integrating traceability into broadcast encryption schemes.

We emphasize here that this method is not specific to the "open one-level" scheme in [9]. Rather, all that is needed to execute this method is a mechanism for assigning the keys in step 2 of Method 1. For example, another scheme such as the "open two-level" scheme in [9] may be used as well.

5.2 Adding Broadcasting Capability to Traceability Schemes

In this section, we take an approach that's similar to the one in Section 5.1, by analyzing how much broadcasting capability is inherent in a traceability scheme. We start by considering some combinatorial properties of both types of schemes. The following lemma is used in [7] and [17] to prove lower bounds on the number of users in a c-traceability scheme.

Lemma 8 ([7,17]). In a c-traceability scheme with users $u_1,...,u_n$, the following must be true:

$$\forall U_{i_1}, ..., U_{i_{c+1}} \text{ distinct, } U_{i_1} \not\subseteq \bigcup_{j=2}^{c+1} U_{i_j}$$

Using the above lemma and Lemma 3, we prove a result on the broadcasting capability inherent in an arbitrary traceability scheme.

Theorem 9. A c-traceability scheme can be used as a broadcast encryption scheme with OR protocols that can exclude any set of m users for any $m \leq c$.

Method 2

Input:

- c-traceability scheme, \mathcal{T}
- an integer, m, the desired number of excluded users (for simplicity, we assume that s = m/c is an integer.)

Output:

- a c-traceability scheme, \mathcal{T}' , which can exclude m users

Construction:

- 1. Let $\{k_1, ..., k_K\}$ be the set of keys in \mathcal{T} . For each j = 1, ..., s, create independent sets of keys $\{k_{j,1}, ..., k_{j,K}\}$. These sets of keys can be viewed as the copies of the scheme \mathcal{T} .
- 2. If a user u has key k_{ℓ} in \mathcal{T} , then in \mathcal{T}' , u is allocated the s keys, $k_{1,\ell},...,k_{s,\ell}$.
- 3. Let E be a set of m users to be excluded and let $P = \{u_1, ..., u_n\} E$ be the set of privileged users. Partition E into s subsets $E_1, ..., E_s$ such that each E_i has size c = m/s. Let $P_i = \{u_1, ..., u_n\} E_i$.
- 4. To broadcast a secret B_P to set P, the center first generates s shares of B_P , $B_P^1, ..., B_P^s$, such that all s shares are necessary to recover B_P (i.e. it's an (s, s)-threshold scheme). Then the center broadcasts B_P^i to the set P_i in accordance with the protocol for P_i in \mathcal{T} (by Theorem 9 this could be an OR protocol).

Table 4. A method for integrating broadcasting capability into traceability schemes.

Proof: If for some $m \leq c$, there exist distinct sets $U_1, ..., U_{m+1}$ such that $U_1 \subseteq \bigcup_{j=2}^{m+1} U_j$ then clearly those sets cannot be part of a c-traceability scheme. The result follows from Lemma 3. \square

Hence, a c-traceability scheme can easily be used to construct a broadcast encryption scheme that can exclude any set of c users. Since the resulting scheme is based on OR protocol, it's also c-resilient.

To achieve more broadcasting capability (i.e. the ability to exclude more users), we need to add more "structure" to the way in which keys are assigned to users. A simple method for accomplishing this is to make "copies" of a single traceability scheme. This method is presented in Table 4.

The traceability of the scheme constructed by Method 2 is inherited from the traceability scheme that is input to the method. Informally, this is because the number of keys per user grows with the number of copies of the original traceability scheme. Therefore, a sufficiently large number of the keys in a decoder must all be contained in one of the copies, and then a traitor tracing algorithm can be applied to those keys.

Theorem 10. Let \mathcal{T} be a traceability scheme with parameters (n, c, K, r, t) and broadcasting capability c. Then there exists a traceability scheme \mathcal{T}' which

has broadcasting capability m and parameters (n, c, K', r', t'), where $K' = \frac{mK}{c}$, $r' = \frac{mr}{c}$, and $t' = \frac{mt}{c}$.

Proof: We first show the broadcasting capability of \mathcal{T}' . Since for any excluded user u, there exists a j such that $u \notin P_j$, u is unable to obtain B_P^j , and hence, u is unable to obtain the message B_P .

To see that \mathcal{T}' has c-traceability, we note that a decoder contains sr keys, where s=m/c. Since there are s copies of \mathcal{T} , one of the copies must contain at least r keys. Hence, the c-traceability of \mathcal{T}' follows from the c-traceability of \mathcal{T} .

All the other assertions about T' follow from Method 2. \square

5.3 Comments on These Methods

For both of the general methods presented here, keys are allocated to users according to a certain matrix. If we look at the key allocations in schemes constructed under either method, the keys appear to be randomly assigned to users along one dimension, but well structured along the other dimension. The random dimension facilitates traceability because it disperses the users' key sets and the structured dimension contributes to broadcasting capability because it indicates which keys to use to exclude different sets of users. Method 1 adds a dimension of randomness to broadcast encryption to achieve high traceability, while Method 2 adds a dimension of structure to traceability schemes to achieve high broadcasting capability. Hence, the two methods can be viewed as complementary to each other.

We also remark that using the new broadcast encryption schemes in Section 4 in conjunction with Method 1, one can construct broadcast encryption schemes with high traceability, high resiliency, and full scalability.

Acknowledgments

We would like to thank Beverly Schmoock, Yuan Ma, Satomi Okazaki and everyone at RSA Labs in San Mateo, CA for their help in preparing this paper. We would also like to thank the anonymous referees for their useful comments.

References

- 1. M. Abdalla, Y. Shavitt and A. Wool. *Towards Making Broadcast Encryption Practical*. To appear in the Proceedings of Financial Cryptography '99, Lecture Notes in Computer Science.
- 2. L. Babai and P. Frankl. Linear Algebra Methods in Combinatorics with Applications to Geometry and Computer Science. Preliminary Version 2, September 1992. Available from the authors.
- 3. S. Berkovits. *How to Broadcast a Secret*. In Advances in Cryptology Eurocrypt '91, Lecture Notes in Computer Science **547** (1992), pp. 536-541.

- C. Blundo and A. Cresti. Space Requirements for Broadcast Encryption. In Advances in Cryptology Eurocrypt '94, Lecture Notes in Computer Science 950 (1994), pp. 287-298.
- 5. C. Blundo, L. A. Frota Mattos and D. Stinson. Trade-offs Between Communication and Storage in Unconditionally Secure Systems for Broadcast Encryption and Interactive Key Distribution. In Advances in Cryptology Crypto '96, Lecture Notes in Computer Science 1109 (1996), pp. 387-400.
- 6. R. Canetti, T. Malkin and K. Nissim. Efficient Communication-Storage Tradeoffs for Multicast Encryption. In Advances in Cryptology Eurocrypt '99, Lecture Notes in Computer Science.
- B. Chor, A. Fiat, and M. Naor. Tracing Traitors. In Advances in Cryptology -Crypto '94, Lecture Notes in Computer Science 839 (1994), pp. 257-270. Final version with B. Pinkas, preprint.
- 8. P. Erdös, P. Frankl and Z. Füredi. Families of Finite Sets in which No Set is Covered by the Union of r Others. Israel Journal of Mathematics **51** (1985), pp.75-89
- 9. A. Fiat and M. Naor. *Broadcast Encryption*. In Advances in Cryptology Crypto '93, Lecture Notes in Computer Science **773** (1994), pp. 480-491.
- M. Just, E. Kranakis, D. Krizanc and P. van Oorschot. On Key Distribution via True Broadcasting. In Proceedings of 2nd ACM Conference on Computer and Communications Security, November 1994, pp. 81-88.
- M. Luby and J. Staddon. Combinatorial Bounds for Broadcast Encryption. In Advances in Cryptology - Eurocrypt '98, Lecture Notes in Computer Science, 1403(1998), pp. 512-526.
- M. Naor and B. Pinkas. Threshold Traitor Tracing. In Advances in Cryptology -Crypto '98, Lecture Notes in Computer Science, 1462 (1998), pp. 502-517.
- J. Staddon. A Combinatorial Study of Communication, Storage and Traceability in Broadcast Encryption Systems. Ph.D. thesis, University of California at Berkeley, 1997.
- 14. D. Stinson. Cryptography: Theory and Practice. CRC Press, 1995.
- 15. D. Stinson. On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption. Designs, Codes and Cryptography 12 (1997), pp. 215-243.
- 16. D. Stinson and T. van Trung. Some New Results on Key Distribution Patterns and Broadcast Encryption. Designs, Codes and Cryptography 14 (1998), pp. 261-279.
- 17. D. Stinson and R. Wei. Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes. SIAM J. Discrete Math, 11 (1998), pp. 41-53.
- D. Stinson and R. Wei. Key Preassigned Traceability Schemes for Broadcast Encryption. In the Proceedings of SAC '98, Lecture Notes in Computer Science, 1556 (1999), pp. 144-156.
- D. Stinson and R. Wei. An Application of Ramp Schemes to Broadcast Encryption. Information Processing Letters 69 (1999), pp. 131-135.
- D. Wallner, E. Harder and R. Agee. Key Management for Multicast: Issues and Architectures. Internet Draft, 1997.

Differential Power Analysis

Paul Kocher, Joshua Jaffe, and Benjamin Jun

Cryptography Research, Inc. 870 Market Street, Suite 1088 San Francisco, CA 94102, USA. http://www.cryptography.com E-mail: {paul,josh,ben}@cryptography.com.

Abstract. Cryptosystem designers frequently assume that secrets will be manipulated in closed, reliable computing environments. Unfortunately, actual computers and microchips leak information about the operations they process. This paper examines specific methods for analyzing power consumption measurements to find secret keys from tamper resistant devices. We also discuss approaches for building cryptosystems that can operate securely in existing hardware that leaks information.

Keywords: differential power analysis, DPA, SPA, cryptanalysis, DES

1 Background

Attacks that involve multiple parts of a security system are difficult to predict and model. If cipher designers, software developers, and hardware engineers do not understand or review each other's work, security assumptions made at each level of a system's design may be incomplete or unrealistic. As a result, security faults often involve unanticipated interactions between components designed by different people.

Many techniques have been designed for testing cryptographic algorithms in isolation. For example, differential cryptanalysis[3] and linear cryptanalysis[8] can exploit extremely small statistical characteristics in a cipher's inputs and outputs. These methods have been well studied because they can be applied by analyzing only one part of a system's architecture — an algorithm's mathematical structure.

A correct implementation of a strong protocol is not necessarily secure. For example, failures can be caused by defective computations[5, 4] and information leaked during secret key operations. Attacks using timing information[7, 11] as well as data collected using invasive measuring techniques[2, 1] have been demonstrated. The U.S. government has invested considerable resources in the classified TEMPEST program to prevent sensitive information from leaking through electromagnetic emanations.

2 Introduction to Power Analysis

Most modern cryptographic devices are implemented using semiconductor logic gates, which are constructed out of transistors. Electrons flow across the sili-

con substrate when charge is applied to (or removed from) a transistor's gate, consuming power and producing electromagnetic radiation.

To measure a circuit's power consumption, a small (e.g., 50 ohm) resistor is inserted in series with the power or ground input. The voltage difference across the resistor divided by the resistance yields the current. Well-equipped electronics labs have equipment that can digitally sample voltage differences at extraordinarily high rates (over 1GHz) with excellent accuracy (less than 1% error). Devices capable of sampling at 20MHz or faster and transferring the data to a PC can be bought for less than \$400.[6]

Simple Power Analysis (SPA) is a technique that involves directly interpreting power consumption measurements collected during cryptographic operations. SPA can yield information about a device's operation as well as key material.

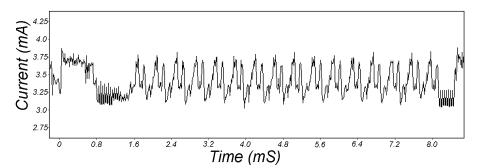


Figure 1: SPA trace showing an entire DES operation.

A trace refers to a set of power consumption measurements taken across a cryptographic operation. For example, a 1 millisecond operation sampled at 5 MHz yields a trace containing 5000 points. Figure 1 shows an SPA trace from a typical smart card as it performs a DES operation. Note that the 16 DES rounds are clearly visible.

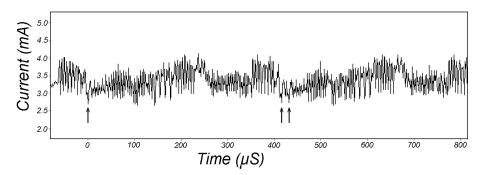


Figure 2: SPA trace showing DES rounds 2 and 3.

Figure 2 is a more detailed view of the same trace showing the second and

third rounds of a DES encryption operation. Many details of the DES operation are now visible. For example, the 28-bit DES key registers C and D are rotated once in round 2 (left arrow) and twice in round 3 (right arrows). In Figure 2, small variations between the rounds just can be perceived. Many of these discernable features are SPA weaknesses caused by conditional jumps based on key bits and computational intermediates.

Figure 3 shows even higher resolution views of the trace showing power consumption through two regions, each of seven clock cycles at 3.5714 MHz. The visible variations between clock cycles result primarily from differences in the power consumption of different microprocessor instructions. The upper trace in Figure 3 shows the execution path through an SPA feature where a jump instruction is performed, and the lower trace shows a case where the jump is not taken. The point of divergence is at clock cycle 6 and is clearly visible.

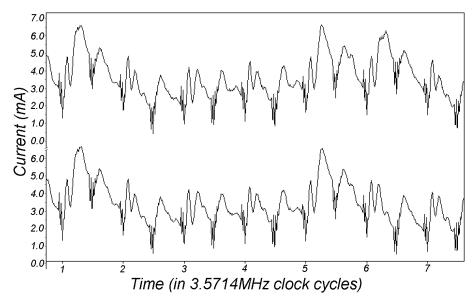


Figure 3: SPA trace showing individual clock cycles.

Because SPA can reveal the sequence of instructions executed, it can be used to break cryptographic implementations in which the execution path depends on the data being processed. For example:

DES key schedule: The DES key schedule computation involves rotating 28-bit key registers. A conditional branch is commonly used to check the bit shifted off the end so that "1" bits can be wrapped around. The resulting power consumption traces for a "1" bit and a "0" bit will contain different SPA features if the execution paths take different branches for each.

DES permutations: DES implementations perform a variety of bit permutations. Conditional branching in software or microcode can cause significant

power consumption differences for "0" and "1" bits.

Comparisons: String or memory comparison operations typically perform a conditional branch when a mismatch is found. This conditional branching causes large SPA (and sometimes timing) characteristics.

Multipliers: Modular multiplication circuits tend to leak a great deal of information about the data they process. The leakage functions depend on the multiplier design, but are often strongly correlated to operand values and Hamming weights.

Exponentiators: A simple modular exponentiation function scans across the exponent, performing a squaring operation in every iteration with an additional multiplication operation for each exponent bit that is equal to "1". The exponent can be compromised if squaring and multiplication operations have different power consumption characteristics, take different amounts of time, or are separated by different code. Modular exponentiation functions that operate on two or more exponent bits at a time may have more complex leakage functions.

3 Preventing SPA

Techniques for preventing simple power analysis are generally fairly simple to implement. Avoiding procedures that use secret intermediates or keys for conditional branching operations will mask many SPA characteristics. In cases such as algorithms that inherently assume branching, this can require creative coding and incur a serious performance penalty.

Also, the microcode in some microprocessors cause large operand-dependent power consumption features. For these systems, even constant execution path code can have serious SPA vulnerabilities.

Most (but not all) hard-wired hardware implementations of symmetric cryptographic algorithms have sufficiently small power consumption variations that SPA does not yield key material.

4 Differential Power Analysis of DES Implementations

In addition to large-scale power variations due to the instruction sequence, there are effects correlated to data values being manipulated. These variations tend to be smaller and are sometimes overshadowed by measurement errors and other noise. In such cases, it is still often possible to break the system using statistical functions tailored to the target algorithm.

Because of its widespread use, the Data Encryption Standard (DES) will be examined in detail. In each of the 16 rounds, the DES encryption algorithm performs eight S box lookup operations. The 8 S boxes each take as input six key bits exclusive-ORed with six bits of the R register and produce four output bits. The 32 S output bits are reordered and exclusive-ORed onto L. The halves L and R are then exchanged. (For a detailed description of the DES algorithm, see [9].)

The DPA selection function $D(C, b, K_s)$ is defined as computing the value of bit $0 \le b < 32$ of the DES intermediate L at the beginning of the 16th round for ciphertext C, where the 6 key bits entering the S box corresponding to bit b are represented by $0 \le K_s < 2^6$. Note that if K_s is incorrect, evaluating $D(C, b, K_s)$ will yield the correct value for bit b with probability $P \approx \frac{1}{2}$ for each ciphertext.

To implement the DPA attack, an attacker first observes m encryption operations and captures power traces $\mathbf{T}_{1..m}[1..k]$ containing k samples each. In addition, the attacker records the ciphertexts $C_{1..m}$. No knowledge of the plaintext is required.

DPA analysis uses power consumption measurements to determine whether a key block guess K_s is correct. The attacker computes a k-sample differential trace $\Delta_D[1..k]$ by finding the difference between the average of the traces for which $D(C, b, K_s)$ is one and the average of the traces for which $D(C, b, K_s)$ is zero. Thus $\Delta_D[j]$ is the average over $C_{1..m}$ of the effect due to the value represented by the selection function D on the power consumption measurements at point j. In particular,

$$\Delta_{D}[j] = \frac{\sum_{i=1}^{m} D(C_{i}, b, K_{s}) \mathbf{T}_{i}[j]}{\sum_{i=1}^{m} D(C_{i}, b, K_{s})} - \frac{\sum_{i=1}^{m} (1 - D(C_{i}, b, K_{s})) \mathbf{T}_{i}[j]}{\sum_{i=1}^{m} (1 - D(C_{i}, b, K_{s}))}$$

$$\approx 2 \left(\frac{\sum_{i=1}^{m} D(C_{i}, b, K_{s}) \mathbf{T}_{i}[j]}{\sum_{i=1}^{m} D(C_{i}, b, K_{s})} - \frac{\sum_{i=1}^{m} \mathbf{T}_{i}[j]}{m} \right).$$

If K_s is incorrect, the bit computed using D will differ from the actual target bit for about half of the ciphertexts C_i . The selection function $D(C_i, b, K_s)$ is thus effectively uncorrelated to what was actually computed by the target device. If a random function is used to divide a set into two subsets, the difference in the averages of the subsets should approach zero as the subset sizes approach infinity. Thus, if K_s is incorrect,

$$\lim_{m \to \infty} \Delta_D[j] \approx 0$$

because trace components uncorrelated to D will diminish with $\frac{1}{\sqrt{m}}$, causing the differential trace to become flat. (The actual trace may not be completely flat, as D with K_s incorrect may have a weak correlation to D with the correct K_s .)

If K_s is correct, however, the computed value for $D(C_i, b, K_s)$ will equal the actual value of target bit b with probability 1. The selection function is thus correlated to the value of the bit manipulated in the 16th round. As a result, the $\Delta_D[j]$ approaches the effect of the target bit on the power consumption as $m \to \infty$. Other data values, measurement errors, etc. that are not correlated to D approach zero. Because power consumption is correlated to data bit values, the plot of Δ_D will be flat with spikes in regions where D is correlated to the values being processed.

The correct value of K_s can thus be identified from the spikes in its differential trace. Four values of b correspond to each S box, providing confirmation of key block guesses. Finding all eight K_s yields the entire 48-bit round subkey. The remaining 8 key bits can be found easily using exhaustive search or by analyzing

one additional round. Triple DES keys can be found by analyzing an outer DES operation first, using the resulting key to decrypt the ciphertexts, and attacking the next DES key. DPA can use known plaintext or known ciphertext and can find encryption or decryption keys.

Figure 4 shows four traces prepared using known plaintexts entering a DES encryption function on another smart card. On top is the reference power trace showing the average power consumption during DES operations. Below are three differential traces, where the first was produced using a correct guess for K_s . The lower two traces were produced using incorrect values for K_s . These traces were prepared using 1000 samples ($m = 10^3$). Although the signal is clearly visible in the differential trace, there is a modest amount of noise.

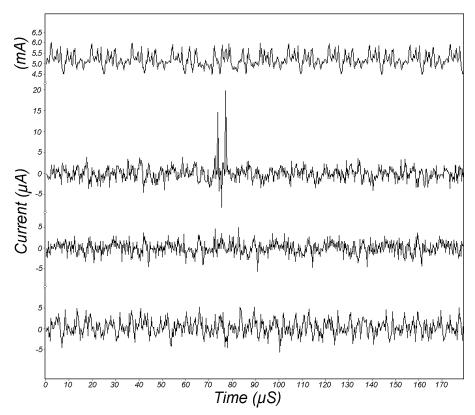


Figure 4: DPA traces, one correct and two incorrect, with power reference.

Figure 5 shows the average effect of a single bit on detailed power consumption measurements. On top is a reference power consumption trace. The center trace shows the standard deviation in the power consumption measurements. Finally, the lower trace shows a differential trace prepared with $m = 10^4$. Note that regions that are not correlated to the bit are more than an order of magnitude closer to zero, indicating that little noise or error remains.

The size of the DPA characteristic is about $40\mu\text{A}$, which is several times less than the standard deviation observed at that point. The rise in the standard deviation at clock cycle 6 coinciding with a strong characteristic indicates that the operand value has a significant effect on the instruction power consumption and that there is considerable variation in the operand values being manipulated. Because low-level instructions often manipulate several bits, a selection function can simultaneously select for values of multiple bits. The resulting DPA characteristics tend to have larger peaks, but do not necessarily have better signal-to-noise ratios because fewer samples are included in the averaging.

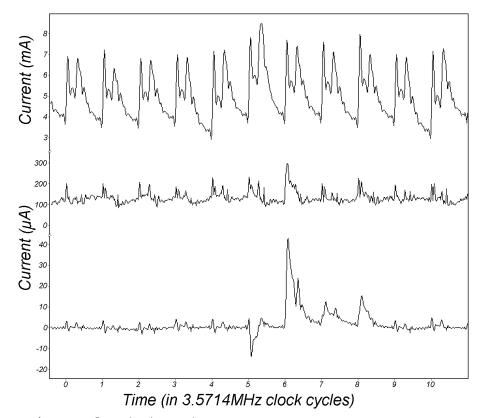


Figure 5: Quantitative DPA measurements

Several sources introduce noise into DPA measurements, including electromagnetic radiation and thermal noise. Quantization errors due to mismatching of device clocks and sample clocks can cause additional errors. Finally, uncorrected temporal misalignment of traces can introduce a large amount of noise into measurements.

Several improvements can be applied to the data collection and DPA analysis processes to reduce the number of samples required or to circumvent countermeasures. For example, it is helpful to correct for the measurement variance,

yielding the significance of the variations instead of their magnitude. One variant of this approach, automated template DPA, can find DES keys using fewer than 15 traces from most smart cards.

More sophisticated selection functions may also be used. Of particular importance are high-order DPA functions that combine multiple samples from within a trace. Selection functions can also assign different weights to different traces or divide traces into more than two categories. Such selection functions can defeat many countermeasures, or attack systems where partial or no information is available about plaintexts or ciphertexts. Data analysis using functions other than ordinary averaging are useful with data sets that have unusual statistical distributions

5 Differential Power Analysis of Other Algorithms

Public key algorithms can be analyzed using DPA by correlating candidate values for computation intermediates with power consumption measurements. For modular exponentiation operations, it is possible to test exponent bit guesses by testing whether predicted intermediate values are correlated to the actual computation. Chinese Remainder Theorem RSA implementations can also be analyzed, for example by defining selection functions over the CRT reduction or recombination processes.

In general, signals leaking during asymmetric operations tend to be much stronger than those from many symmetric algorithms, for example because of the relatively high computational complexity of multiplication operations. As a result, implementing effective SPA and DPA countermeasures can be challenging.

DPA can be used to break implementations of almost any symmetric or asymmetric algorithm. We have even used the technique to reverse-engineer unknown algorithms and protocols by using DPA data to test hypotheses about a device's computational processes. (It may even be possible to automate this reverse-engineering process.)

6 Preventing DPA

Techniques for preventing DPA and related attacks fall roughly into three categories.

A first approach is to reduce signal sizes, such as by using constant execution path code, choosing operations that leak less information in their power consumption, balancing Hamming Weights and state transitions, and by physically shielding the device. Unfortunately such signal size reduction generally cannot reduce the signal size to zero, as an attacker with an infinite number of samples will still be able to perform DPA on the (heavily-degraded) signal. In practice, aggressive shielding can make attacks infeasible but adds significantly to a device's cost and size.

A second approach involves introducing noise into power consumption measurements. Like signal size reductions, adding noise increases the number of samples required for an attack, possibly to an infeasibly-large number. In addition, execution timing and order can be randomized. Designers and reviewers must approach temporal obfuscation with great caution, however, as many techniques can be used to bypass or compensate for these effects. Several vulnerable products have passed reviews that used naïve data processing methods. For safety, it should be possible to disable temporal obfuscation methods during review and certification testing.

A final approach involves designing cryptosystems with realistic assumptions about the underlying hardware. Nonlinear key update procedures can be employed to ensure that power traces cannot be correlated between transactions. As a simple example, hashing a 160-bit key with SHA[10] should effectively destroy partial information an attacker might have gathered about the key. Similarly, aggressive use of exponent and modulus modification processes in public key schemes can be used to prevent attackers from accumulating data across large numbers of operations. Key use counters can prevent attackers from gathering large numbers of samples.

Using a leak-tolerant design methodology, a cryptosystem designer must define what leakage rates and functions that the cryptography can survive. Leakage functions can be analyzed as oracles providing information about computational processes and data, where the leakage rate is the upper bound on the amount of information provided by the leakage function. Implementers can then use leak reduction and leak masking techniques as needed to meet the specified parameters. Finally, reviewers must verify that the design assumptions are appropriate and correspond to the physical characteristics of the completed device.

7 Related Attacks

Electromagnetic radiation is a particularly serious issue for devices that pass keys or secret intermediates across a bus. Even a simple A.M. radio can detect strong signals from many cryptographic devices. A wide variety of other signal measurement techniques (such as superconducting quantum imaging devices) also show promise. Statistical methods related to SPA and DPA can be used to find signals in noisy data.

8 Conclusions

Power analysis techniques are of great concern because a very large number of vulnerable products are deployed. The attacks are easy to implement, have a very low cost per device, and are non-invasive, making them difficult to detect. Because DPA automatically locates correlated regions in a device's power consumption, the attack can be automated and little or no information about the target implementation is required. Finally, these attacks are not theoretical

or limited to smart cards; in our lab, we have used power analysis techniques to extract keys from almost 50 different products in a variety of physical form factors.

The only reliable solution to DPA involves designing cryptosystems with realistic assumptions about the underlying hardware. DPA highlights the need for people who design algorithms, protocols, software, and hardware to work closely together when producing security products.

References

- 1. R. Anderson, M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," Security Protocol Workshop, April 1997, http://www.cl.cam.ac.uk/ftp/users/rja14/tamper2.ps.gz.
- R. Anderson and M. Kuhn, "Tamper Resistance a Cautionary Note", The Second USENIX Workshop on Electronic Commerce Proceedings, November 1996, pp. 1-11.
- E. Biham and A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.
- E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," Advances in Cryptology: Proceedings of CRYPTO '97, Springer-Verlag, August 1997, pp. 513-525.
- D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," Advances in Cryptology: Proceedings of EURO-CRYPT '97, Springer-Verlag, May 1997, pp. 37-51.
- Jameco Electronics, "PC-MultiScope (part #142834)," February 1999 Catalog, p. 103
- 7. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," Advances in Cryptology: Proceedings of CRYPTO '96, Springer-Verlag, August 1996, pp. 104-113.
- 8. M. Matsui, "The First Experimental Cryptanalysis of the Data Encryption Standard," Advances in Cryptology: Proceedings of CRYPTO '94, Springer-Verlag, August 1994, pp. 1-11.
- National Bureau of Standards, "Data Encryption Standard," Federal Information Processing Standards Publication 46, January 1977.
- National Institute of Standards and Technology, "Secure Hash Standard," Federal Information Processing Standards Publication 180-1, April 1995.
- J. Dhem, F. Koeune, P. Leroux, P. Mestré, J. Quisquater, and J. Willems, "A practical implementation of the timing attack," UCL Crypto Group Technical Report Series: CG-1998/1, 1998.
- R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, 21, 1978, pp. 120-126.

Towards Sound Approaches to Counteract Power-Analysis Attacks

Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi

I.B.M. Thomas J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A. Email: {schari,csjutla,jrrao,roha}@watson.ibm.com

Abstract. Side channel cryptanalysis techniques, such as the analysis of instantaneous power consumption, have been extremely effective in attacking implementations on simple hardware platforms. There are several proposed solutions to resist these attacks, most of which are ad-hoc and can easily be rendered ineffective. A scientific approach is to create a model for the physical characteristics of the device, and then design implementations provably secure in that model, i.e, they resist generic attacks with an a priori bound on the number of experiments. We propose an abstract model which approximates power consumption in most devices and in particular small single-chip devices. Using this, we propose a generic technique to create provably resistant implementations for devices where the power model has reasonable properties, and a source of randomness exists. We prove a lower bound on the number of experiments required to mount statistical attacks on devices whose physical characteristics satisfy reasonable properties.

1 Introduction

Side channel cryptanalysis i.e., cryptanalysis using information leaked during the computation of cryptographic primitives has successfully been used to break implementations on simple platforms such as chip-cards[1, 2, 8]. It has been claimed [2] that in chip-card like devices, all straightforward implementations are susceptible to attack by power analysis techniques. Analogous to timing attacks[1], simple power attacks, where the adversary extracts key bits by identifying the execution sequence from the instantaneous power consumption, are easier to protect against by making the execution sequence independent of the key bits. Differential and Higher order Differential power attacks, on the other hand are extremely powerful and difficult to protect against. These attacks rely on the ability of the attacker to create two different statistical distributions on the values being manipulated during a single instruction (or a set of instructions) based on known input/output and guesses of few key bits. If these distributions can be distinguished, using statistical tests on instantaneous power samples (or any other side channel), then the attacker can verify the key guesses.

Due to the wide-ranging impact of these attacks, there have been several proposed commercial implementations which claim to resist these and similar attacks. Without rigorous justification, several of these solutions are ad-hoc

and based on simplistic techniques such as probabilistically reordering execution paths. It is our view that such approaches essentially miss the import of these attacks and their underlying basis. Furthermore, these simple countermeasures can be nullified by signal processing. Instead, we propose that the focus should be on sound scientific approaches to this problem i.e. develop an accurate and abstract model of the problem and identify rigorously proven techniques which can be used as effective countermeasures. A realistic goal for a sound and effective countermeasure is to provably resist all generic attacks by an adversary who is allowed to perform and observe at most an a priori fixed number of adaptively chosen operations. Generic attacks are those that use functional specifications and generic physical characteristics and do not depend on specific implementations and devices. Typically, simple single chip devices and the keys within them are short-lived and the number of key-dependent operations performed by them over their lifetime is also limited. A countermeasure secure against an a priori bound on the number of experiments, can be used in practice by explicitly enforcing this upper bound in the devices themselves.

In this paper, we propose a general, simplified model for the power consumption in simple devices, and use this to restate the basis of these attacks and to analyze countermeasures. We examine ad-hoc approaches which have been proposed and discuss why they are easy to defeat. A general technique is then proposed as a countermeasure against statistical attacks in devices where the power model is reasonable and a source of randomness is available. This technique is based on well known secret sharing schemes where each bit of the original computation is divided probabilistically into shares such that any proper subset of shares is statistically independent of the bit being encoded and thus, yields no information about the bit. Computation of cryptographic primitives is done accessing only the random shares at each point, with intermediate steps computing only the shares of the result. Splitting the bit into multiple probabilistic shares amplifies the uncertainty of the adversary at each point and forces him to work with the joint distributions of the signal at the points where the shares are being accessed. For computation of common cryptographic primitives, simple sharing schemes based on XOR and addition modulo 28 can be used.

We make realistic assumptions about the power consumption model for devices with respect to the uncertainty of the adversary at each point, and analyze the efficacy of this technique to withstand power analysis attacks. Using this, we rigorously prove lower bounds on the number of observations required to statistically distinguish distributions, defined in power attacks, using observations on power samples. Our lower bounds are exponential in the number of shares that each bit (or byte) of the computation is encoded by. The models and lower bounds are initial steps in developing a formal framework for the problem of computation in the presence of the information leaked due to the observations on the physical characteristics of devices. Only solutions which can be proved secure in a formalized model should be considered for implementation. Substantial effort is still required to find more appropriate models and stronger analysis.

Section 2 describes a formalization of a general power consumption model.

Section 2.1 restates power analysis attacks in this framework. In section 3, we analyze countermeasures, examine simple ad-hoc solutions which are ineffective, and propose the secret sharing scheme as a general countermeasure against these attacks. In Section 3.4, we make realistic approximations on the model and rigorously prove lower bounds on the number of samples needed to mount differential power attacks against implementations with this countermeasure.

2 Power Model and Definitions

CMOS devices consume power only when changes occur in logic states, while no significant power is needed to maintain a state. Examples of changes include changes in the contents of the RAM, internal registers, bus-lines, states of gates and transistors etc. In simple chips almost all activity is triggered by an internal/external clock edge and all activity ceases well before the next clock edge. A few processes, such as on-chip noise generators, operate independently of the clock and consume a small, possibly random amount of power continuously. Each clock edge triggers a sequence of power consuming events within the chip, as dictated by the microcode, bringing it to the next state. This sequence depends on parts of the current state of the processor and parts of the state of other subsystems accessed in that cycle. We define relevant state bits as the bits of the overall state which determine the sequence of events, and hence the power, during a clock cycle. Depending on the cycle, the relevant state bits could include bits of internal registers, bits on internal and external buses, address bits and contents of memory locations being accessed etc.

The instantaneous power consumption of the chip shortly after a clock edge is a combination of the consumption components from each of the events that have occurred since the clock edge. Each event's timing and power consumption depends on physical and environmental factors such as the electrical properties of the chip substrate, layout, temperature, voltage etc., as well as coupling effects between events of close proximity. As a first approximation, we ignore coupling effects and create a linear model, i.e., we assume that the power consumption function of the chip is simply the sum of the power consumption functions of all the events that take place.

Consider a particular cycle of a particular instruction in the execution path of some fixed code. At the start of the cycle, the chip is in one of several relevant states (determined by the value of the relevant state bits) depending on the input and processing done in earlier cycles. Let $\mathcal S$ denote the set of possible relevant states when control reaches this cycle and let $\mathcal E$ be the set of all possible events that can occur in a cycle. For each $s \in \mathcal S$, and each $e \in \mathcal E$, let occurs(e,s) be the binary function which is 1 if e occurs when the relevant state is s and 0 otherwise. Let delay(e,s) be the time delay of the occurrence of event e in state s from the clock edge and let f(e,t) denote the power consumption impulse function of event e with respect to time t (t=0 when e occurs and f(e,t)=0 for t<0). In our linear model, P(s,t), the power consumption function of the chip in that

cycle with state s and time t after the clock edge can be written as

$$P(s,t) = \sum_{e \in \mathcal{E}} f(e,t-delay(e,s)) * occurs(e,s)$$

Due to the presence of noise and asynchronous power consuming components, a better model is:

$$P(s,t) = \mathcal{N}_c(t) + \sum_{e \in \mathcal{E}} (f(e,t-delay(e,s) + \mathcal{N}_d(e,s)) + \mathcal{N}(e,t)) * occurs(e,s)(1)$$

where $\mathcal{N}(e,t)$ is a Gaussian noise component associated with the power consumption function of e, $\mathcal{N}_d(e,s)$) is a Gaussian noise component affecting the delay function and \mathcal{N}_c is the Gaussian external noise component.

2.1 Statistical Power Attacks

Equation 1 shows the strong dependence between the power consumption function and the relevant state which is the basis of all statistical power attacks. Let P_1 and P_2 be two different probability distributions on the relevant state before the clock edge of a certain cycle. From equation 1, it is very likely that the distribution of the instantaneous power when the state is drawn from P_1 will be different from the distribution of the instantaneous power when the state is drawn from P_2 . This difference and the distinguishability of different distributions by statistical tests on power samples, is the basis for Differential Power attacks (DPA). Simple distributions are sufficient to mount these attacks. For example, in the DPA attacks described in [2], P_1 and P_2 are very simple: P_1 is the uniform distribution on the set of all relevant states which have a particular relevant state bit 1 and P2 is the uniform distribution on the set of all relevant states which have the same bit 0. The difference in the power distribution for these two cases represents the effect of that particular relevant state bit on the net power consumption. This can be used to extract cryptographic keys by guessing parts of keys, using this to predict a relevant state bit and defining the distributions as described above. Higher order differential attacks are those in which the distributions P_1 and P_2 are defined over multiple internal state variables and where the adversary has access to multiple side channels. Appendix 1 shows an example of distributions induced in the power consumption signal by distributions on the relevant state, for an actual chip-card.

In defining security against statistical attacks we use the strongest possible notion: using the side channel, with high probability, the adversary should not be able to predict with even a slight advantage, any bit that he could not predict from just the knowledge of inputs, outputs and program code. In using the side channel, the adversary is limited to trying to distinguish distributions which he can affect by choice of inputs and selection based on outputs. These are limited to distributions on bits such as bits in the algorithm specification e.g., bits of the key, bits which depend directly on the key and the input etc. Also, in an attack against the implementation the adversary could affect deterministic temporary

variables, registers etc. We informally define the set of realizable distributions which the adversary can directly affect as follows:

Definition 1. A distribution on state bits is realizable if the adversary can induce the distribution by suitable choice of inputs and selection based on outputs. In particular, this excludes distributions of state bits which result from explicit randomization introduced in the implementation outside of the specification.

The ability to distinguish any two realizable distributions is potentially advantageous to the adversary. Using standard notations (see for example[7]) we define the distinguishing probability of an adversary as follows:

Definition 2. Let M be a binary valued adversary who adaptively chooses k inputs and has access to the side channel signals for the corresponding operations. Let B_1 and B_2 be any two realizable distributions on the bits of a computation, and D_1 and D_2 the distributions induced on the side channel signals, by the choice of inputs and B_1 and B_2 respectively. Let M^D denote M's output when given k input/output pairs and corresponding side-channel samples from a distribution D. The distinguishing probability of M when given samples from distributions D_1 and D_2 is $|\Pr(M^{D_1} = 1) - \Pr(M^{D_2} = 1)|$. M is said to distinguish B_1 from B_2 using k side-channel samples, if the distinguishing probability of M, on D_1 and D_2 , is at least some constant c.

Using this definition of adversaries, we define a secure computation. We intend to capture *extra* information that the adversary obtains from the side channel.

Definition 3. A computation is said to be secure against N sample side channel cryptanalysis, if for all adversaries M and all realizable distributions B_1 and B_2 , if M can distinguish B_1 from B_2 using fewer than N samples, then M can distinguish B_1 and B_2 without the side channel.

The attacks described by [2] can be restated as using the side channel to distinguish distributions B_1 and B_2 which correspond to almost uniform distributions on a few relevant state bits, with a particular state bit (depending on the input and key) being 0 and 1 respectively. There the adversary bases its decision by comparing the mean of the given samples, with some known threshold.

3 Countermeasures to Power Analysis

Using these formal definitions of side channel cryptanalysis, we discuss general countermeasures against such attacks. First, we examine several ad-hoc approaches to fixing this problem, which, we believe, miss the import of these attacks and can easily be rendered ineffective. We present a probabilistic encoding scheme with which we can effectively perform secure computations. Based on realistic approximations of the power models of Section 2 we prove lower bounds on the number of samples required to distinguish distributions.

3.1 Ad-hoc Approaches

Due to the commercial impact, several ad-hoc solutions are currently being implemented and claim to be resistant to these statistical attacks. Unfortunately, most can be defeated by signal processing in conjunction with only moderately more samples. Allowing for about 1 million possible experiments, it is reasonable to assume that the adversary can exploit every relevant state bit in any instruction to mount a statistical attack, provided he can efficiently predict that bit in a significant fraction of the runs based on the code specification, known inputs and small number of guesses for parts of the key.

Some approaches to protecting computation use simple countermeasures such as "balancing", i.e., try to negate the effects of one set of events by another "complementary" set. For example, by ensuring that all bytes used in computation have Hamming weight 4, one can try to negate the effect of each 1 bit by a corresponding 0 bit. Such approaches fail at high resolution and large number of samples, because the power consumption functions and timing of two "complementary" events will be slightly different and the adversary can maximize these differences by adjusting the operating conditions of the card. Another popular approach is to randomize the execution sequence i.e. keep operations the same, but permute the order e.g. in DES, the S-boxes are looked up in a random order. Unless this random sequencing is done extensively throughout the computation, which may be impossible since the specification forces a causal ordering, it can be undone and a canonical order re-created by signal processing. Attacks can be mounted on the re-ordered signals. Even if the entire computation cannot be canonically reordered, it is sufficient to identify "corresponding" sample points in different runs so that a significant fraction are samples from the same power function P for the same cycle. All statistical attacks that work for P are also applicable to "corresponding" points, although more samples would be needed due to "noise" introduced by unrelated samples. In the case of permuted S-boxes, if the permutation is random, in $\frac{1}{8}$ of the runs S-box 1 is looked up first, and in the remaining samples, the signal at this point, corresponding to different lookups, is essentially random. Thus, even with no reordering, we now have a signal which is attenuated by a factor of 8. Mounting the original attack with 64 times the number of samples yields the same results. Elementary reordering substantially reduces this factor. A similar countermeasure in hardware is typically achieved by making instructions take a variable number of cycles or by having the cycles be of varying length (see [4]). Once again, it is very easy to negate all these countermeasures with signal processing.

3.2 A general countermeasure

A general countermeasure is to ensure that the adversary cannot predict any relevant bit in any cycle, without making run-specific assumptions independent of the actual inputs to the computation. This makes statistical tests involving several experiments impossible, since the chance of the adversary making the correct assumptions for each run is extremely low. While this yields secure

computation, it is not clear how one can do effective computation under this requirement since no bit depending directly on the data and key can be manipulated at any cycle. In some cases the function being computed has algebraic properties that permits such an approach, e.g., for RSA one could use blinding [1, 3] to partially hide the actual values being manipulated. Another class of problems where this is possible is the class of random self-reducible problems [9]. Such structure is unlikely to be present in primitives such as block ciphers.

3.3 Encoding

The encoding we propose is to randomly split every bit of the original computation, into k shares where each share is equiprobably distributed and every proper subset of k-1 shares is statistically independent of the encoded bit. Computation can then be carried securely by performing computation only the shares, without ever reconstructing the original bit. Shares are refreshed after every operation involving them to prevent information leakage to the adversary.

To fix a concrete encoding scheme, we assume that each bit is split into k shares using any scheme which has the required stochastic properties. For instance, bit b can be encoded as the k shares $b \oplus r_1, r_2, \ldots, r_{k-1}, r_1 \oplus \ldots \oplus r_{k-1}$, where the r_i s are randomly chosen bits. Furthermore, assume that each share is placed in a separate word at a particular bit position and all other bits of the share word are chosen uniformly at random.

In practice, it would be more useful, if each word of computation is split similarly into k shares. In that case, other schemes of splitting into shares based on addition mod 2^8 , subtraction mod 2^8 would also be viable. Encoding bytes of data manipulated by splitting them into shares would yield the optimal performance. Ignoring the initial setup time, the performance penalty in performing computation using just the k shares is a factor of k. Our results which have been proved based on the bit encoding scheme would also work for this case but the bounds they yields are based only on the characteristics of the noise within the chip, and hence may not be optimal. This is discussed briefly after the analysis for the bit encoding case. The results and analysis we present here can serve as a framework in which to prove results for the byte encoding scheme.

The method to encode the bit in secret shares should be chosen based on the computation being protected. For instance, for an implementation of DES, the XOR scheme is ideal since the basic operations used are XOR, permutations, and table lookups. Table lookups can be handled by first generating a random rearrangement of the original table since a randomized index will be used to look up the table. This step increases the overhead beyond the factor of 2.

In practice, the splitting technique needs to be applied only for a sufficient number of steps into the computation until the adversary has very low probability of predicting bits, i.e., till sufficient secret key dependent operations have been carried out. Similar splitting also has to be done at end of the computation if the adversary can get access to its output. For instance, in DES, one needs to use the splitting scheme only for the first four and last four rounds.

3.4 Analysis

We analyze the encoding scheme described above, by making reasonable assumptions on distribution of side channel information and prove that the amount of side channel information required grows exponentially in k, the number of shares. For concreteness we fix the XOR bit encoding scheme and consider the instantaneous power consumption at some time instant in a cycle manipulating a share. The relevant state in that cycle will not only include a share of the bit, but also all the other random bits in the word. It is quite reasonable to assume that the contributions of all bits in the word will be similar in magnitude. From equation (1), expanding occurs(e, s) as a linear form over the bits of s, the instantaneous power consumption when a particular share is being manipulated will be

$$P = b \times s_0 + P \times s_0 + R$$

where b is the contribution of just the shared bit s_0 , $P \times s_0$ is the distribution of power contributions of events which require s_0 and other state bits and finally, R is the distribution of events which are independent of the bit s_0 . In operations such as load, store and XOR, if s_0 is a bit in a word being manipulated, the factor P can be viewed as a small perturbation on the real value b. In simple operations there is no "interaction" between the different bits of the value being manipulated and an approximation, we will ignore the contribution of the variable P. The random variable R is typically much larger than b since it includes the sum of similar contributions from all other bits. For most operations, R is the sum of almost independent distributions which is very well approximated by the Normal Distribution. Thus, we make the realistic assumption, which has been empirically tested as shown in Appendix 1, that R has a normal distribution with mean μ and variance σ^2 . The results we prove can also be shown to hold in the case that R is the sum of i.i.d's, which is the case for operations such as load, store, XOR. Further work needs to be done to analyze more complex and precise distributions which model all chip-card operations such as multiply where there is interaction between the bits being manipulated and it is unlikely that one can ignore the contribution of the variable P.

Assume that in each sample the adversary has access to the k signal values corresponding to the power consumption at instructions which access the shares $b\oplus r_1, r_2, \ldots, r_{k-1}, r_1\oplus r_2\ldots\oplus r_{k-1}$. Rewrite these bits as $r_1, \ldots r_k$, with $r_1\oplus\ldots\oplus r_k=b$. Denote the distribution of the instantaneous power consumption signal at these points by random variables $Z_1, Z_2, \ldots Z_k$. Also, let $Z_i=A_i+X_i$, where A_i is the contribution due to the bit in concern and X_i is the additive factor which follows the distribution R. By the definition of the encoding, A_i takes values 0 and 1 with probability $\frac{1}{2}$ each. Any noise in the contribution due to A_i can be absorbed in R without affecting the distribution of R since R is typically much bigger than b. Thus, the power contribution due to A_i is 1 if $r_i=1$ (and 0 if $r_i=0$). It is important to note that the A_i 's are not independent since $A_1\oplus\ldots\oplus A_k=b$.

In defining distributions that an adversary can try to distinguish using inputs, outputs and the side channel information, note that the adversary can not control

the randomizing variables r_i 's. Thus, the only realizable distributions are those with the value of the bit b being 0 and 1 *i.e.* distributions D_1 and D_2 where: a random variable Y sampled from D_1 is given by

$$\langle A_1 + X_1, \ldots, A_k + X_k \rangle$$

with the condition that $A_1 \oplus ... \oplus A_k = b$, while a random variable sampled from D_2 is given by

$$\langle A_1 + X_1, \ldots, A_k + X_k \rangle$$

with the condition that $A_1 \oplus \ldots \oplus A_k = 1 \oplus b$. This is more general than the encoding scheme specified above. It corresponds to the intuition that using \oplus and k random bits there are several ways to split a bit into shares e.g. a three way split of a bit b can be specified as $b \oplus r_1$, r_2 , $r_1 \oplus r_2$ or as $b \oplus r_1 \oplus r_2$, r_1 , r_2 .

Let M be an adversary trying to distinguish the two distributions D_1 and D_2 . It gets a sequence T of m samples, sampled from either D_1 or D_2 , each element of which is a k tuple of signal values at the k points that the shares are accessed. If S_1, \ldots, S_k are random variables denoting these values let $S = (S_1 - \mu) \times \cdots \times (S_k - \mu)$ where μ is the mean of the distribution R. S has a slightly different mean (with the difference of $\frac{1}{2^{(k-1)}}$) under distributions D_1 and D_2 and with a variance of approximately $(\sigma^2)^k$. Using standard techniques, it is easy to show that an adversary given $(2\sigma^2)^k$ samples can distinguish the two distributions using the statistic S. Thus, approximately n^k samples are sufficient, where $n = \sigma^2$. We are interested in lower bounds on the number of samples required to distinguish the distributions. Our central result is:

Theorem 4. Let δ be a constant. Given distributions D_1 and D_2 defined above, any adversary which has access to $m < n^{\frac{k}{2}-4\delta}$ samples $(n = \sigma^2)$ from one of these two distributions, has probability at most $n^{-\delta}$ of distinguishing D_1 and D_2 .

Note that this not a tight lower bound and we conjecture that n^k is the tight bound. We sketch the proof for the case k=2 and the general proof can be done along the same lines. We require the following basic facts from probability theory.

3.5 Probability Theory Basics

The density function of Normal distribution with mean μ and variance σ^2 is

$$\eta(x)=rac{1}{\sqrt{2\pi}\sigma}e^{-rac{1}{2}(rac{x-\mu}{\sigma})^2}$$

The corresponding distribution function is defined as

$$N(x)=\int_{-\infty}^{x}\eta(x)dx$$

The following inequality is useful (see for example [5]):

$$N(x) \ < \ rac{\sigma^2}{\mu - x} \eta(x)$$

Theorem 5. Chernoff Bound: Let $S_n = X_1 + ... + X_n$ where the X_i s are independent and are 1 and 0 with probability p and q = 1 - p. For p < a < 1, and b = 1 - a,

$$\Pr[S_n \ge na] \approx e^{-nK(a,p)}$$

where $K(a, p) = a \log(\frac{a}{p}) + b \log(\frac{b}{q})$.

3.6 Lower Bound for 2 way Split

In this section we outline a proof of Theorem 4 with k=2. Our proof uses several techniques and ideas from Naor et. al [6]. The realizable distributions D_1 and D_2 which the adversary has to distinguish are defined on the space $\mathcal{P}=\mathcal{R}\times\mathcal{R}$, where \mathcal{R} is the set of reals. If Y is the random variable sampled from one of these distributions, in D_1 , $Y=\langle A+X_1,A+X_2\rangle$ and $Y=\langle A+X_1,(1-A)+X_2\rangle$, where X_1,X_2 are random variables with normal distribution with parameters μ , σ and A is a uniform binary random variable.

Let M be an adversary trying to distinguish D_1 and D_2 . By assumption, M fixes a certain precision ϵ and divides the area $\mathcal{R} \times \mathcal{R}$ into squares of length ϵ , where $\epsilon < \frac{1}{\sqrt{n}}$ without loss of generality. All inputs in a particular square are treated identically. When we refer to $\langle u, v \rangle \in \mathcal{R} \times \mathcal{R}$, we identify u and v with the boundaries of the intervals containing them and thus identify $\langle u, v \rangle$ with the the $\epsilon \times \epsilon$ square containing it.

Let $m = n^{1-4\delta}$, where δ is a constant. We show that no adversary can distinguish between sequences with at most m samples, sampled according to D_1 and D_2 . In the following exposition, T is a random variable denoting a randomly drawn sequence, and s denotes a possible value of T. The outline of the proof is as follows: We first define a set of bad sequences (definition 6 below). Then we show (in Lemma 8) that under distributions D_1 and D_2 the probability that a sampled sequence T is bad i.e. the probability of the event BAD_T is very small. Restricting ourselves to sequences which are not bad, we show that the probability that the random variable T has a particular value s is almost the same whether we are sampling according to D_1 or D_2 . In particular, in Lemma 10 we show that $\Pr_{D_1}(T=s|\neg BAD_T) > \mu_n * \Pr_{D_2}(T=s|\neg BAD_T)$, where μ_n is close to 1, from above. Similarly, we show that the probability of a sequence which is not bad, when sampled according to D_2 is at least μ_n^{-1} times its probability under D_1 . In other words, the occurrence probability of a sequence that is not bad, is almost the same under both distributions. Putting it all together, we then show that the adversary cannot distinguish the distributions using fewer than m samples. We begin with the definition of bad sequences.

Definition 6. Let $f_s(x,y), x,y \in \mathcal{R}$, be the number of times that $\langle x,y \rangle$ appears in sequence s. We call a sequence s a bad sequence if either (1) $f_s(\mu-u,\mu-v)>0$, for $u,v>n^{(0.5+\delta)}$ or (2) $f_s(\mu-u,\mu-v)>\frac{n^2}{(uv+1)}\cdot n^{-c}$, for other values, u,v>0. Here $c=1-3\delta$.

In the above definition and in the rest of the proof we have ignored the cases when u, v < 0 and these can be treated symmetrically.

Definition 7. Define $maxf(\mu - u, \mu - v) = \theta$, for $u, v > n^{(0.5+\delta)}$ and $maxf(\mu - u, \mu - v) = \frac{n^2}{(uv+1)} \cdot n^{-c}$, for all other values $u, v \geq 0$.

This the maximum possible number of times that $\langle u, v \rangle$ occurs in a sequence which is not bad. Denote the random sequence of m two tuples as $T = T_1 T_2 ... T_m$ and denote $s = s_1 s_2 ... s_m$.

Lemma 8. $\Pr_{D_1,D_2}(BAD_T) < e^{-n^{\delta}}$ i.e. under either distribution, the set of bad sequences is negligibly small.

Proof: We consider the two cases in the definition of a bad sequence separately. The probability that the random variable distributed according to $N(\mu, \sigma)$ takes on a particular value $\mu - x$ is given by $N(\mu - x) - N(\mu - x - \epsilon)$, which for small values of ϵ can be approximated by $\epsilon \cdot \eta(\mu - x)$. Using this approximation and taking into account the contribution of the binary valued random variable, under either distribution D_1 or D_2 the probability that $s_i = \langle \mu - u, \mu - v \rangle$ can be approximated by probability

$$p = \frac{d\epsilon^2}{(2\pi\sigma^2)} \cdot e^{-\frac{1}{2}(\frac{u^2 + v^2}{\sigma^2})}$$
 (2)

where d is a small constant close to 1. Since the elements of the sequence are sampled independently, by the Chernoff bound (section 3.5), the probability P_{uv} that $f_s(\mu-u,\mu-v) > \frac{n^2}{(uv)} \cdot n^{-c}$ is about $e^{-m \cdot K(a,p)}$, where $a = \frac{n^{-c}}{(uv+1)}$. Since 1-a is close to 1, K(a,p), a simple calculation shows that

$$P_{uv} \approx (\frac{p}{a})^{ma} \cdot (\frac{1-p}{1-a})^{m} < (\frac{d\epsilon^{2}uvm}{2\pi\sigma^{2}n^{2-c}})^{\frac{n^{2-c}}{uv}} \cdot e^{n^{1-c}} < (\frac{\epsilon^{2}e}{n^{2\delta}})^{(\frac{n^{2-c}}{(uv+1)})}$$

Since there are $\frac{\sqrt{n^2}}{\epsilon^2}$ possible values of u, v, the total probability of BAD in case (2) is at most

$$\frac{n}{\epsilon^2} \cdot \epsilon^{2(\frac{n^{2-c}}{(uv+1)})}$$

which is exponentially small as $uv < n^{1+2\delta}$.

For case (1) of the definition of bad sequences, let s_i be the two tuple $\langle s_{i1}, s_{i2} \rangle$. For each i, using the inequality on Normal distribution in Section 2,

$$N(\mu - n^{0.5 + \delta}) \ < \ rac{\sigma^2}{n^{0.5 + \delta}} e^{-rac{1}{2}(rac{n^{0.5 + \delta}}{\sigma})^2}$$

Thus $\Pr(s_{i1} < \mu - n^{0.5 + \epsilon(n)}) < e^{-(n^{2\delta} - \log n)}$. The probability that the sequence is bad according to case (1) is at most m times this small probability. \square Thus the space of bad sequences is very small. We now argue that for sequences that are not bad, the probability of occurrence is the same under both distributions. Denote $\Pr_{D_1}(T_i = \langle \mu - u, \mu - v \rangle)$, by $X_{u,v}$. Also, let $\Pr_{D_2}(T_i = \langle \mu - u, \mu - v \rangle)$ be $X_{u,v} + \Delta_{u,v}$. The difference $\Delta_{u,v}$ is due to the contribution of the binary valued random variable. The following lemma bounds $\Delta_{u,v}$.

Lemma 9. For small ϵ , $\frac{\Delta_{u,v}}{X_{u,v}} \leq \frac{uv}{\sigma^4}$.

Proof: This can be seen through the following sequence of approximations and identities. The first approximation follows from the definition of D_1 and D_2 and by choice of small ϵ .

$$\begin{split} \Delta_{u,v} &\approx \frac{1}{2} \cdot (\eta(\mu - u)\eta(\mu - v) + \eta(\mu - u - 1)\eta(\mu - v - 1) \\ &- \eta(\mu - u - 1)\eta(\mu - v) - \eta(\mu - u)\eta(\mu - v - 1)) \cdot \epsilon^2 \\ &= \frac{1}{2} \cdot \epsilon^2 \cdot (\eta(\mu - u) - \eta(\mu - u - 1)) \cdot (\eta(\mu - v) - \eta(\mu - v - 1)) \\ &= \frac{\epsilon^2}{2} \cdot \frac{1}{(\sqrt{2\pi}\sigma)^2} \cdot (e^{-\frac{1}{2}(\frac{u}{\sigma})^2} - e^{-\frac{1}{2}(\frac{u+1}{\sigma})^2}) \cdot (e^{-\frac{1}{2}(\frac{v}{\sigma})^2} - e^{-\frac{1}{2}(\frac{v+1}{\sigma})^2}) \\ &\leq \frac{\epsilon^2}{2} \cdot \frac{1}{2\pi\sigma^2} \cdot (\frac{u}{\sigma^2} \cdot e^{-\frac{1}{2}(\frac{u}{\sigma})^2}) \cdot (\frac{v}{\sigma^2} \cdot e^{-\frac{1}{2}(\frac{v}{\sigma})^2}) \\ &\leq \frac{d}{2} \cdot X_{u,v} \cdot \frac{uv}{\sigma^4} \end{split}$$

The second last inequality follows from the power series expansion of e^x . The last inequality and the constant d are from (2). Thus the claim follows.

Lemma 10. The probability of occurrence of a sequence that is not bad is almost the same under both distributions. In particular, $\mu_n^{-1} * \Pr_{D_1}(T = s | \neg BAD_T) < \Pr_{D_2}(T = s | \neg BAD_T) < \mu_n * \Pr_{D_1}(T = s | \neg BAD_T)$, where $\mu_n = 1 + (2n)^{-\delta}$.

Proof: We just show that

$$|\mathit{Pr}_{D_2}(T=s|\neg \mathit{BAD}_T) - \mathit{Pr}_{D_1}(T=s|\neg \mathit{BAD}_T)| < \mathit{Pr}_{D_1}(T=s|\neg \mathit{BAD}_T) * n^{-c\sqrt{n}}.$$

This follows by:

$$\begin{split} &|Pr_{D_{2}}(T=s|\neg BAD_{T}) - Pr_{D_{I}}(T=s|\neg BAD_{T})| \\ &= |\Pi_{u,v}(X_{u,v} + \Delta_{u,v})^{f_{s}(u,v)} - \Pi_{u,v}(X_{u,v})^{f_{s}(u,v)}| \\ &= \Pi_{u,v} X_{u,v}^{f_{s}(u,v)} \cdot |(\Pi_{u,v}(1+\Delta_{u,v}/X_{u,v})^{f_{s}(u,v)}) - 1| \\ &\leq \Pi_{u,v} X_{u,v}^{f_{s}(u,v)} \cdot |(\Pi_{u,v}(1+\Delta_{u,v}/X_{u,v})^{maxf(u,v)}) - 1| \\ &\leq \Pi_{u,v} X_{u,v}^{f_{s}(u,v)} \cdot |(\Pi_{u,v}(e^{n^{-c}})) - 1| \\ &\leq \Pi_{u,v} X_{u,v}^{f_{s}(u,v)} \cdot |((e^{n^{-c}})^{m}) - 1| \\ &\leq \Pi_{u,v} X_{u,v}^{f_{s}(u,v)} \cdot |(e^{n^{-b}}) - 1| \\ &\leq (\Pi_{u,v} X_{u,v}^{f_{s}(u,v)}) \cdot (2n)^{-\delta} \\ &= Pr_{D_{1}}(T=s|\neg BAD_{T}) \cdot (2n)^{-\delta} \end{split}$$

The first two equalities are by definition 6 and the fact that $(1+x)^{1/x} < e$, for x > 0. Although the number of u, v pairs can be about (n/ϵ^2) , the third inequality is true because the number of u, v for which $f_s(u, v) > 0$ is at most

m. The fifth inequality follows by the power series expansion of e^x , from which it can be shown that for x < 1, $e^x < 1 + 2x$.

Proof (of Theorem 4) We put together the various pieces to show that an adversary M cannot distinguish these two distributions. Let M be a binary valued adversary and let M(s) denote the output of M on input s, a sequence of samples. Note that if C is any condition on the random variables, by definition

$$Pr(M^D = 1 \mid C) = \Sigma_s Pr_D(T = s \mid C) \cdot Pr(M(s) = 1) \text{ and } T = s)$$
 (3)

By definition,

$$\begin{split} |Pr(M^{D_1} = 1) - Pr(M^{D_2} = 1)| &= \\ |Pr(M^{D_1} = 1| \neg BAD_T) * Pr_{D_I}(\neg BAD_T) \\ &- Pr(M^{D_2} = 1| \neg BAD_T) * Pr_{D_I}(\neg BAD_T) \\ &+ Pr(M^{D_1} = 1| BAD_T) * Pr_{D_I}(BAD_T) \\ &- Pr(M^{D_2} = 1| BAD_T) * Pr_{D_I}(BAD_T)| \\ &\leq |Pr(M^{D_1} = 1| \neg BAD_T) * (Pr_{D_I}(\neg BAD_T) - \delta_n * Pr_{D_I}(\neg BAD_T))| \\ &+ |Pr(M^{D_1} = 1| BAD_T) * Pr_{D_I}(BAD_T) \\ &- Pr(M^{D_2} = 1| BAD_T) * Pr_{D_I}(BAD_T)| \end{split}$$

where $\mu_n^{-1} \leq \delta_n \leq \mu_n$. This follows from the observation (3) and Lemma 10. Since μ_n is close to one, and $\Pr_{D_1,D_2}(\neg BAD_T)$ is also close to one, the first summand on the right of the above inequality is close to zero. The second summand is also close to zero as $\Pr_{D_1,D_2}(BAD_T)$ is close to zero by lemma 8. Thus, the distinguishing probability is close to zero.

Theorem 11. Let D_1 and D_2 be as before but with the noise being the sum of n identically and independently distributed binary variables (with $p = \frac{1}{2}$). Any adversary which has access to $n^{k/2-4\epsilon}$ samples, has probability at most $\frac{1}{n^{\epsilon}}$ of distinguishing D_1 and D_2 .

3.7 Encoding Bytes

For practical computation, we would use the encoding scheme of splitting each relevant byte of the computation into k shares. It is clear from our proof techniques that if there was enough additional noise in the power signals to effectively mask the byte values, then the same proofs will go through for the byte encoding scheme. It seems unlikely to happen in limited devices. It may be possible to extend our proof techniques to account for the fact that there is uncertainty on the value of a byte being manipulated given its power signal even without any additional noise.

4 Conclusions and Directions

We have presented a simplified initial step into the formal analysis of computing in the presence of loss of entropy due to leaked side channel information. Our lower bounds on the amount of side channel information required are proved for reasonable approximations of the actual distributions. Substantial effort is required to find more effective and general countermeasures against such attacks. Besides proving implementations secure from power attacks, this framework could also be used to design ciphers and other primitives which readily admit a secure, efficient implementation.

Acknowledgements

The authors thank D. Coppersmith, H. Scherzer, S. Weingart, M. Witzel and E. Yashchin for fruitful discussions about several issues presented in this paper.

References

- P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. Advances in Cryptology-Crypto '96, Lecture Notes in Computer Science # 1109, pp 104-113.
- 2. P. Kocher, J. Jaffe and B. Jun. Differential Power Analysis: Leaking Secrets. In *Proceedings of Crypto '99*. A preliminary version is available online at http://www.cryptography.com/dpa/technical/index.html.
- 3. D. Chaum. Blind Signatures for Untraceable Payments. Advances in Cryptology: Proceedings of Crypto '82, Plenum Press, 1983, pp 199-203.
- J. Daemen and V. Rijmen. Resistance against implementation attacks: A comparative study of the AES proposals. Proceedings of the Second AES Candidates Conference, Mar 1999, Rome, Italy.
- W. Feller. An introduction to Probability Theory and its application, Vol. 1, Wiley Mathematical Statistics Series, 1950.
- M. Naor, O. Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited. In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, pp 189-199.
- M. Luby. Pseudorandomness and cryptographic applications. Princeton University Press.
- 8. The complete unofficial TEMPEST web page. Available at http://www.eskimo.com/~joelm/tempest.html.
- 9. M. Abadi, J. Feigenbaum, and J. Kilian. On Hiding Information from an Oracle. Journal of Computer and System Sciences, 39(1):21-50, Aug. 1989.

6 Appendix 1: Power distribution example

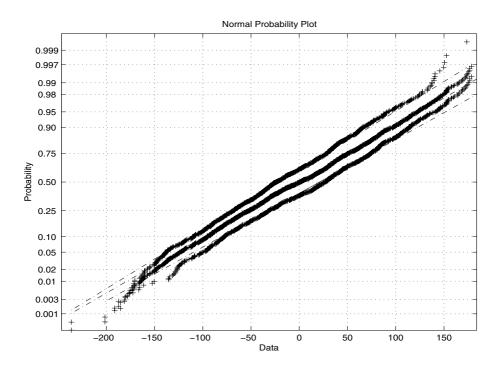


Fig. 1. Power distributions when loading random bytes from RAM

Figure 1 shows three distinct distributions of the instantaneous power consumption of a commonly available chip in the middle of a cycle which loads the value of a RAM byte into the accumulator. These correspond to three different distributions on the value of that particular RAM byte. All three power distributions are plotted on a "normal scale" and each distribution shows up as a thick line in this plot, which means all these three power distributions are close to normal. The middle line corresponds to the power distribution when the RAM byte is drawn uniformly at random. It has a mean of 0 (we have shifted all power readings by an additive constant to enforce this). The top line corresponds to the power distribution when the RAM byte is uniformly chosen from all bytes with MSB of 1. It has a mean of -25. The bottom line corresponds to the power distribution when the RAM byte is uniformly chosen from all bytes with MSB 0. This has a mean of +25.

Separability and Efficiency for Generic Group Signature Schemes

(Extended Abstract)

Jan Camenisch*

BRICS**

Department of Computer Science University of Aarhus Ny Munkegade DK - 8000 Århus C, Denmark camenisch@ieee.org Markus Michels

Entrust Technologies Europe r3 security engineering ag Glatt Tower, 6th floor CH – 8301 Glattzentrum, Switzerland

Markus.Michels@entrust.com

Abstract. A cryptographic protocol possesses separability if the participants can choose their keys independently of each other. This is advantageous from a key-management as well as from a security point of view. This paper focuses on separability in group signature schemes. Such schemes allow a group member to sign messages anonymously on the group's behalf. However, in case of this anonymity's misuse, a trustee can reveal the originator of a signature. We provide a generic fully separable group signature scheme and present an efficient instantiation thereof. The scheme is suited for large groups; the size of the group's public key and the length of signatures do not depend on the number of group member. Its efficiency is comparable to the most efficient schemes that do not offer separability and is an order of magnitude more efficient than a previous scheme that provides partial separability. As a side result, we provide efficient proofs of the equality of two discrete logarithms from different groups and, more general, of the validity of polynomial relations in \mathbb{Z} among discrete logarithms from different groups.

1 Introduction

Multiparty cryptographic protocols are typically preceded by a setup phase in which the involved entities choose their public and secret keys besides other system-parameters. Often their choices depend on the other entities' choices. Not only forces this the entities to choose new key-pairs for each (kind of) protocol they participate in but also to re-choose their keys if other entities renew theirs or if the system-parameters are changed. Apart from such key-management problems, a dependency between different keys can cause security problems as well. For instance, if some entity choose weak parameters or discloses his/her secret keys, other entities' security can be weakened as well.

 $^{^{\}star}$ Current address: IBM Zurich, Säumerstrasse 4, CH-8803 Rüschlikon.

^{**} Basic Research in Computer Science, Center of the Danish National Research Foundation.

To overcome such problems the concept of separability introduced in [30] is promoted and refined. A cryptographic protocol is said to enjoy perfect separability if all entities can chose not only their keys independently of the other entities but also any instance of the required cryptographic primitives, e.g., any existentially unforgeable signature scheme. A protocol has strong separability if the entities are restricted in their choice of the instances of the primitives, e.g., only the RSA [43] signature scheme with a 1024 bit modulus, and it possesses weak separability if their choices need to depend on some system parameters, e.g., the generator of some prime order algebraic group. Hence, weak separability does not overcome potential key-management problems but is satisfactory from a security point of view. Perfect, and to some extent also strong separability, is a prerequisite that a cryptographic scheme can be set-up in absence of all potentially involved parties: it suffices that their public keys are authentically available (and satisfy certain requirements in case of strong separability). Furthermore, a cryptographic protocol can only have partial separability of any kind, that is, separability only with respect to some of the participating entities. At the cost of prohibitive inefficiency perfect separable protocols can in most cases be obtained by combining different cryptographic primitives with general, (e.g., circuit-based) zero-knowledge proof techniques.

In this paper we consider group signature schemes and show how perfect separability can be achieved by providing a generic but potentially inefficient solution. We then restrict ourselves to strong separability and point out how efficiency can be gained. A group signature scheme allows group-members to sign messages anonymously and unlinkably on behalf of the group. To counterfeit misuse of this anonymity the scheme enables a third party, called revocation manager, to reveal the identity of a signature's originator. Since its introduction by Chaum and van Heyst [17], a number of researchers have proposed more efficient solutions and diversified the model. The first schemes presented [8,17,18,40] have the property that the length of signatures and/or the size of the group's public key depend on the size of the group and thus they are not suited for large groups. This drawback is overcome by the schemes presented in [9,11] as well as by a further one in $[12]^1$. The less efficient scheme of [11] was shown to be insecure but can easily be adjusted [2]. However, none of these schemes enjoys separability since the revocation manager's keys depend on the ones chosen by the membership manager.

A concept dual to group signature schemes is identity escrow [30]. It can be regarded as a group identification scheme with revocable anonymity. In fact, any identity escrow scheme with a 3-move identification protocol can be turned into a group signature scheme by applying the Fiat-Shamir heuristic [23] to the identification protocol; the opposite is achieved by signing a random message and then proving the knowledge of a signature on the chosen message. The schemes

Other schemes having the same properties are either completely broken [31,38] as shown in [33,34] or do not satisfy the usual security requirements (i.e., the scheme proposed in [1] is not secure against colluding group members while in the protocol given in [32] signatures are linkable).

presented in [30] are partially separable (according to our nomenclature), i.e., strongly separable with respect to the revocation manager but not with respect to group members. Due to their construction, they are suitable for large groups as well but are much less efficient compared to the schemes presented in [9,11,12]. The schemes presented in this paper possess strong separability with respect to all participants and their efficiency is in the same order of magnitude as the ones in [9,11,12].

As a side result, we present efficient proofs of the validity of polynomial relations in \mathbb{Z} among discrete logarithms from *different* groups, which might be of independent interest, in particular, for obtaining separable schemes for other cryptographic scenarios.

2 A Model of Separable Group Signature Schemes

This section describes the model of separable group signature schemes and states the security requirements. The main difference to the model of ordinary group signature schemes is that here the key generation of the membership manager, the revocation manager, and the group members are individual procedures that are independent of each other. We assume only a single revocation manager and membership manager, but the definition extends easily to several of them.

Definition 1. Let ℓ_M , ℓ_R , and ℓ_U be security parameters. A separable group signature scheme consists of the following procedures:

- GKG-MM: A probabilistic algorithm that on input 1^{ℓ_M} outputs the membership manager's secret key x_M and public key y_M .
- GKG-RM: A probabilistic algorithm that on input 1^{ℓ_R} outputs revocation manager's secret key x_R and public key y_R .
- GKG-GM: A probabilistic algorithm that on input 1^{ℓ_U} outputs a group member's secret key x_U and public key y_U .
- GKG-S: A probabilistic algorithm that generates the system parameter y_S and an empty group member list GML. This list is a public board which anybody can read but only the membership manager and a potential group member U together can add entries related to U's identity.
- Reg: A probabilistic interactive protocol between the group member U and the membership manager. Their common input is the group member's identity ID_U and public key y_U . If both parties accept, their common output is the membership certificate s_U on y_U . Finally, the two parties add y_U and ID_U to the (public) group member list GML.
- GSig: A probabilistic algorithm that on input s_U , x_U , the group's public key Y, and a message m outputs a group signature s on m.
- GVer: An algorithm that on input the group's public key Y, an alleged signature s, and a message m outputs 1 if and only if the signature is valid w.r.t. Y.
- GTrace: An algorithm which on input the revocation manager's secret key x_R , the group's public key Y, a message m, and a signature s on m outputs the identity

 ID_U of the originator of the signature and a proof V that ID_U is indeed the originator.

The group's public key consists of the triple $Y = (y_R, y_M, y_S)$. The following security requirements must hold:

Correctness of signature generation: All signatures on any message generated by any honest group member using GSig will get accepted by the verification algorithm.

Anonymity and unlinkability and of signatures: Given two signature-message pairs, it is only feasible to the revocation manager to find out which group member(s) generated any of the signatures or whether the signatures have been generated by the same group member.

Unforgeability of signatures: It is feasible to sign messages only to group members (i.e., users that have run the registration protocol with the membership manager) or to the membership manager herself².

Unforgeability of tracing: The revocation manager cannot accuse a group member falsely of having originated a given signature.

No framing: No coalition of group members, the revocation manager, and the membership manager can produce a signature that will be associated to a group member not part of the coalition.

Unavoidable traceability: No coalition of group members and the revocation manager (but excluding the membership manager) can generate a valid signature that, when its anonymity is revoked, cannot be associated to a group member.

To achieve strong or perfect separability it is necessary that the four algorithms GKG-MM, GKG-RM, GKG-GM, and GKG-S can be run completely independent of each other. Moreover, if one of them is re-run the others need not.

The actual algorithm to set-up the group signature scheme is GKG-S and should be carried out by representatives of the group members and the membership manager(s) or by a single party they all fully trust. The other three key-generation algorithms are in principle key-generation algorithms of some cryptographic primitives and could have been run in advance of the set-up of the group signature scheme, i.e., they could in principle be excluded from the model.

3 Preliminaries and Proof Techniques

This section summarizes various known results on proofs of knowledge of and about discrete logarithms, combines them into new building blocks, and provides notation for such protocols. In particular, we present new protocols to prove the equality of two discrete logarithms in *different* groups and, more general, the

 $^{^2}$ The membership manager can always invent a fake identity, i.e., run $\mathsf{GKG}\text{-}\mathsf{GM},$ and issue a certificate on that public key. It is understood that if the revocation lead to a public key not present in GML then the membership manager is accused.

validity of polynomial relations in \mathbb{Z} among discrete logarithms from different groups.

In the following we assume different groups $G = \langle g \rangle, G_i = \langle g_i \rangle$ (i = 1, 2) of large known prime orders q and q_i , respectively. Furthermore, let h and h_i also be generators of G and G_i , respectively, such that $\log_g h$ and $\log_{g_i} h_i$ are not known. For technical reasons, we define a discrete logarithm $\log_g h$ to be the integer x with -q/2 < x < q/2 such that $y = g^x$ and assume that arithmetic modulo q is carried out with this in mind. All protocols exposed in the following are two-party protocols between a male prover and a female verifier.

3.1 Basic Zero-Knowledge Proofs of Knowledge

The most basic protocol is a zero-knowledge proof of knowledge of the discrete logarithm of some group element $y \in G$ to the base g [15,45]. We shortly recall this protocol and its properties: The prover knowing $x = \log_g y$ sends the verifier the commitment $t := g^r$, where $r \in_R \mathbb{Z}_q$. Then, the verifier sends the prover a random challenge $c \in_R \{0,1\}^k$ to which the prover responds with s := r - cx (mod q). (The integer $k \geq 1$ is a security parameter.) The verifier accepts if $t = g^s y^c$. Triples (t, c, s) with $t = g^s y^c$ are called accepting triples. Since this $x = \log_g y$ can be computed from two accepting triples (t, c, s) and (t, \dot{c}, \dot{s}) with $c \neq \dot{c}$, i.e., $x := (s - \dot{s})(\dot{c} - c)^{-1}$ (mod q), this protocol is a proof of knowledge of $\log_g y$ when sequentially repeated sufficiently many times. Furthermore, the protocol is honest-verifier zero-knowledge³ and for $k = \mathcal{O}(\log \log q)$ zero-knowledge for any verifier. Using notation from [11], this protocol is denoted $PK\{(\alpha) : y = g^{\alpha}\}$, which can be read as "zero-knowledge Proof of Knowledge of a value α such that $y = g^{\alpha}$ holds." The convention is that Greek letters denote the knowledge proven while all other parameters are known to the verifier.

This basic protocol can be extended to prove the knowledge of a representation of a group element $y \in G$ with respect to several bases z_1, \ldots, z_v [15] which is denoted $PK\{(\alpha_1, \ldots, \alpha_v) : y = z_1^{\alpha_1} \cdot \ldots \cdot z_v^{\alpha_v}\}$. Another extension allows to prove the equality of the discrete logarithms of two group elements $y_1, y_2 \in G$ to the bases g and h, respectively [14,16]. The idea is to carry out the basic protocol for y_1 and for y_2 in parallel and requiring that the challenges (c) and the responses (s) are the same [16]. If the verifier accepts, the two logarithms must be equal as can be seen by considering how the prover's secret can be derived from two accepting triples with the same commitment. Such a protocol will be denoted $PK\{(\alpha) : y_1 = g^{\alpha} \land y_2 = h^{\alpha}\}$. This technique can be generalized to prove equalities among representations of the elements y_1, \ldots, y_w to bases g_1, \ldots, g_v [11]. As an example, the protocol $PK\{(\alpha, \beta) : y_2 = g^{\alpha} \land y_3 = g^{\beta} \land y_1 = y_2^{\beta}\}$ allows to prove that the discrete logarithm of y_1 is the product of the discrete logarithms of y_2 and y_3 modulo the group's order.

These kinds of proofs of knowledge (PK) can be turned into signature schemes by the so-called Fiat-Shamir heuristic [23]. That is, the prover determines the

³ Honest verifier zero-knowledge proofs can be made zero-knowledge by requiring the verifier to commit to the challenge before she receives the prover's commitments.

challenge c by applying a collision-resistant hash-function \mathcal{H} to the commitment and the message m that is signed and then computes the response as usual. The resulting signature consists of the challenge and the response. We denote such Signature schemes based on a zero-knowledge Proof of Knowledge (SPK) similarly as the PK's, e.g., $SPK\{(\alpha): y=g^{\alpha}\}(m)$. Such SPK's can be proven secure in the random oracle model [4,41] given the security of the underlying PK's.

3.2 Interval Proofs for Discrete Logarithms

If we restrict ourselves to binary challenges, i.e., set k=1, the basic protocol $PK\{(\alpha):y=g^{\alpha}\}$ can be modified to prove not only the knowledge of $\log_g y$ but also that $\log_g y$ lies within some determined interval [9,13], e.g., $-2^{\ell} < \log_g y < 2^{\ell}$. More precisely, this is achieved by requiring the prover's response s to satisfy $-2^{\ell-1} < s < 2^{\ell-1}$. However, the prover can only carry out the protocol successfully, if the tighter bound $-2^{(\ell-2)/\epsilon} < \log_g y < 2^{(\ell-2)/\epsilon}$ holds, where e>1 is a security parameter⁴, and when choosing the value r to compute the commitment t such that $-2^{\ell-2} < r < 2^{\ell-2}$. This modified protocol is denoted $PK^b\{(\alpha):y=g^{\alpha} \land (-2^{\ell} < \alpha < 2^{\ell})\}$, where the b in PK^b reminds that the protocol uses binary challenges, i.e., is not very efficient. A further modification allows to prove that $a-2^{\ell} < \log_g y < a+2^{\ell}$, where a is a fixed offset [9]⁵. Note that such protocols make sense only if the group's order q is larger than $2^{\ell+1}$.

An alternative but less efficient method for proving bounds on a discrete logarithm is to first commit to every bit of $x = \log_g y$ and then to prove that (1) they constitute the binary representation of x and (2) the committed values are either a 0 or a 1. The latter can be done using techniques from [20,44]. This method is linear in the length of x but allows to prove tighter bounds and, when revealing the most significant bit as 1, even the exact bit-length of x.

3.3 Efficient Interval Proofs for Discrete Logarithms

The choice k = 1 is not very attractive from an efficiency point of view and thus the quest for more efficient protocols seems natural. Indeed, one can do better under the strong-RSA assumption [3,25] which is the following variation of the well-known RSA assumption [43].

Assumption 1 (Strong RSA). There exists a probabilistic algorithm K such that $Pr[z \stackrel{G}{=} u^e \land e > 1 : (G, z) := K(1^{\ell_g}), (u, e) := A(G, z)] < 1/p(\ell_g)$ holds for all probabilistic polynomial-time algorithms A, all polynomials $p(\cdot)$, all sufficiently large ℓ_g , where G is a group of order $\approx 2^{\ell_g}$, $z \in G$, and $e \in \mathbb{Z}$.

In words, this assumption states that there exists a key-generator K that outputs a group G of unknown order and an element $z \in G/\{\pm 1\}$ such that it is infeasible

⁴ In fact, the parameter ϵ controls the tightness of the statistical zero-knowledgeness. ⁵ Using different techniques, Damgård obtains a similar result [21].

to find a pair $(u, e) \in G \times \mathbb{Z}$ such that e > 1 and $u^e = z$. K could be implemented by choosing G as \mathbb{Z}_n , where n is an RSA modulus of size $\approx 2^{\ell_g}$, and picking z randomly from \mathbb{Z}_n .

Fujisaki and Okamoto [25] show that under this assumption the protocol $PK\{(\alpha): y=g^{\alpha}\}$ works also for groups of unknown order, e.g., for $G=\mathbb{Z}_n^*$, where n is an RSA modulus. More precisely, it is pointed out that the discrete logarithm x of y to the base g can be computed from two accepting triples (t, c, s) and (t, \dot{c}, \dot{s}) without knowing the group's order: Since $t=g^sy^c=g^{\dot{s}}y^{\dot{c}}$ we must have $x(c-\dot{c})\equiv \dot{s}-s\pmod{\mathrm{ord}(g)}$. If $c-\dot{c}$ does not divide $\dot{s}-s$ in \mathbb{Z} , then one can compute a non-trivial root of g (see [25]). However, due to the strong-RSA assumption the latter is infeasible and hence $c-\dot{c}$ must divide $\dot{s}-s$ in \mathbb{Z} with overwhelming probability and we can compute x without knowing the order of g. This argument generalizes to representations w.r.t. several bases.

The fact that $c - \dot{c}$ divides $\dot{s} - s$ in \mathbb{Z} with overwhelming probability allows us to draw conclusions about the size of the prover's secret from the size of his responses [9,13] similarly as in case of binary challenges. In combination with the ideas that led to the protocol $PK\{(\alpha): y_1 = g^{\alpha} \land y_2 = h^{\alpha}\}$ (cf. Section 3.1), this technique can also be used to efficiently prove statements about the size of discrete logarithms from groups with known orders: Let ℓ denote a length, $\epsilon > 1$ and k be security parameter, and $G = \langle g \rangle$ be a group whose order $q > 2^{\ell+1}$ is known. Let the prover's secret x be an integer such that $-2^{(\ell-2)/\epsilon-k} < x < 1$ $2^{(\ell-2)/\epsilon-k}$ and let $y=g^x$. First, the prover and the verifier engage in a (once and for all) set-up phase. The verifier randomly chooses two sufficiently large safe primes, say p_1 and p_2 , and computes $n := p_1 p_2$. (The modulus n must be large enough to avoid factoring but its size needs not to depend on ϵ , ℓ , k, or q.) She chooses two random elements h_1 and h_2 from \mathbb{Z}_n , sends the prover n, h_1 , and h_2 , and proves him that n is indeed the product of two safe primes (e.g., by using the techniques from [10]). The prover checks whether $h_1 \neq \pm 1 \pmod{n}$, $h_2 \neq \pm 1$ \pmod{n} , $\gcd(h_1, n) = 1$, and $\gcd(h_2, n) = 1$ holds. This will convince him that h_1 and h_2 have large order (see [26]). This concludes the set-up phase. Now, the prover chooses $r \in_R \mathbb{Z}_n$, computes $\tilde{y} = h_1^r h_2^x$, and sends the verifier \tilde{y} . Finally, they engage in the protocol $PK\{(\alpha,\beta): y \stackrel{G}{=} g^{\alpha} \wedge \tilde{y} \stackrel{\mathbb{Z}_n^*}{=} h_1^{\beta} h_2^{\alpha} \wedge (-2^{\ell} < \alpha < 2^{\ell})\}.$ If they finish the protocol successfully, the verifier will be convinced that the prover knows a value, say x, such that $y = g^x$ and $-2^{\ell} < x < 2^{\ell}$ holds.

3.4 Proving Relations among Discrete Logs from Different Groups

The protocols for proving that a discrete logarithms lies within some determined interval, as exposed in the previous two subsections, are a powerful tool. For instance, as is shown in [10], they allow to prove that a discrete logarithm is the product (or sum) of two other discrete logarithms in \mathbb{Z} , i.e., not modulo the group's order. This extends naturally to the validity of arbitrary polynomial equations over \mathbb{Z} in the prover's secrets. Moreover, these protocols allow to prove the equality of discrete logarithms from different groups as is explained in the next paragraph. We believe this is a new building block of independent interest,

which in combination with the results from [10] allows to prove the validity of polynomial relations (in \mathbb{Z}) among discrete logarithms from different groups.

Let $G_1 = \langle g_1 \rangle$ and $G_2 = \langle g_2 \rangle$ be two distinct groups of orders q_1 and q_2 , respectively, and let ℓ be a integer such that $2^{\ell+1} < \min\{q_1,q_2\}$ holds. Let $y_1 = g_1^x$ and $y_2 = g_2^x$. If x lies in between $-2^{(\ell-2)/\epsilon}$ and $2^{(\ell-2)/\epsilon}$, where e > 1 is a security parameter (cf. Section 3.2), the prover can convince the verifier that $\log_{g_1} y_1 = \log_{g_2} y_2$ (in \mathbb{Z}) by carrying out $PK^b\{(\alpha): y_1 \stackrel{G_1}{=} g_1^\alpha \land y_2 \stackrel{G_2}{=} g_2^\alpha \land (-2^\ell < \alpha < 2^\ell)\}$ with her. Since this protocol uses binary challenges it is not very efficient. However, under the strong RSA assumption and if x lies in between $-2^{(\ell-2)/\epsilon-k}$ and $2^{(\ell-2)/\epsilon-k}$, where k denotes the number of bits of the challenge, the prover can efficiently convince the verifier that $\log_{g_1} y_1 = \log_{g_2} y_2$ holds as follows. First the prover and the verifier engage in the (once and for all) set-up phase to generate the modulus n and the elements h_1 and h_2 as described in the previous subsection. Then the prover chooses $r \in_R \mathbb{Z}_n$, computes $\tilde{y} = h_1^r h_2^x$ (mod n), sets it to the verifier, and carries out $PK\{(\alpha,\beta): y_1 \stackrel{G_1}{=} g_1^\alpha \land y_2 \stackrel{G_2}{=} g_2^\alpha \land \tilde{y} \stackrel{\mathbb{Z}^*}{=} h_1^\beta h_2^\alpha \land (-2^\ell < \alpha < 2^\ell)\}$ together with the verifier. We describe this protocol in detail:

- 1. The prover picks $r_1 \in_R \{-2^{\ell-2}, \dots, 2^{\ell-2}\}$ and $r_2 \in_R \{-(n2^k)^{\epsilon}, \dots, (n2^k)^{\epsilon}\}$ and computes the commitments $t_1 := g_1^{r_1}, \ t_2 := g_2^{r_1}, \ t_3 := h_1^{r_2}h_2^{r_1}$. He sends the verifier (t_1, t_2, t_3) .
- 2. The verifier returns a random challenge $c \in_R \{0,1\}^k$.
- 3. The prover computes the responses $s_1 = r_1 cx$ and $s_2 = r_2 cr$ (both in \mathbb{Z}) and sends the verifier (s_1, s_2) .
- 4. The verifier accepts if and only if $-2^{\ell-1} < s_1 < 2^{\ell-1}$, $t_1 = g_1^{s_1} y_1^c$, $t_2 = g_2^{s_1} y_2^c$, and $t_3 = h_1^{s_2} h_2^{s_1} \tilde{y}^c$ hold.

Clearly, this kind of protocol generalizes to several different groups, to representations, and to arbitrary modular relations.

4 A Generic Separable Group Signature Scheme

This section provides the definitions of some cryptographic primitives and then presents a generic separable group signature scheme based on these primitives.

4.1 Definition of Some Cryptographic Primitives

The first primitive is a shadow encryption scheme (SE). It consists of three algorithms EKG, Enc, and Dec for key generation, encryption, and decryption, respectively. Participants are a sender and a receiver. On input of a security parameter, EKG outputs the key pair (x, y) of the receiver. On input of the receiver's public key y and a message m, Enc outputs the ciphertext c. On input of the public key y, the secret key x, and the ciphertext c, Dec outputs a value v, called the shadow. Unlike as for ordinary public key encryption schemes, decryption does not reveal the encrypted message m but only a value v such that the pair (m, v) satisfies a predefined binary relation \mathcal{R} .

Definition 2. Let ℓ be a security parameter and $\mathcal{R} \subseteq \{0,1\}^{\ell} \times \{0,1\}^*$ a binary one-to-one relation that can be recognized in polynomial time. A triple (EKG, Enc, Dec) of probabilistic polynomial-time algorithms is a secure publickey shadow encryption scheme with respect to \mathcal{R} if the following properties hold.

Correctness: For every $(x, y) \in EKG(1^{\ell})$ and all $m_1, m_2 \in \{0, 1\}^{\ell}$ we have $(m_1, Dec(y, x, Enc(y, m_2))) \in \mathcal{R}$ if and only if $m_1 = m_2$.

Security: For all probabilistic polynomial-time algorithms T and M, all polynomials $p(\cdot)$, and all sufficiently large ℓ we have

$$\Pr[T(1^{\ell}, y, m_0, m_1, d) = i : (x, y) := EKG(1^{\ell}); (m_0, m_1) := M(y, 1^{\ell});$$
$$i \in_R \{0, 1\}; d := Enc(y, m_i) \} < 1/2 + 1/p(\ell).$$

Furthermore, as encryption and decryption should be efficient, for all messages m values v such that $(m,v) \in \mathcal{R}$ must be efficiently computable. Any semantically secure public key encryption scheme (e.g., [29,37]) will give an shadow encryption scheme with basically the same efficiency. However, if \mathcal{R} is a hard relation (i.e., given v it is infeasible to find an m such that $(m,v) \in \mathcal{R}$), the converse seems to be possible only at a loss of efficiency, e.g., by shadow-encrypting every bit of the message separately. Hence, a shadow encryption scheme is a weaker primitive from an efficiency point of view but is sufficient in applications where full encryption is not necessary. A concept similar to shadow encryption is confirmer commitments [35].

The rest of the primitives are rather standard and therefore we introduce them only informally (for formal definitions see, e.g., [28]). Let f be a one-way function. Let SIG = (SKG, Siq, Ver) denote a signature scheme, where SKG is the key-generation algorithm (that on input 1^{ℓ} outputs a key pair (x,y)), Sig is the signing algorithm (that on input of a secret key x, the corresponding public key y, and a message m outputs a signature s on m), and Ver is the verification algorithm (that on input of a public key y, an alleged signature s, and a message m outputs 1 if and only if s is a signature on m with respect to y). We require that the signature scheme is existentially unforgeable under a certain kind of chosen-message attack, that is the attacker gets only signatures on messages of which he knows a pre-image under some predetermined one-way function f. Finally, we need an unconditionally hiding commitment scheme, i.e., a function Com that takes as input a string x to commit to and a random string r. One can commit to a value x by C := Com(x, r), where r is randomly chosen. We require that the distribution of C committing to different x's are statistically indistinguishable, i.e., x is information theoretically hidden from the receiver. Furthermore, it should be hard to open a commitment in two ways, i.e., hard to find strings $x, x' \neq x, r$, and r' such that Com(x, r) = Com(x', r')

4.2 A Generic Realization of a Separable Group Signature Scheme

This section describes a generic separable group signature scheme and shows its security. The construction extends ideas from [11,12]. Let SE = (EKG, Enc, Dec)

be a probabilistic shadow encryption scheme, SIG = (SKG, Sig, Ver) a signature scheme, f a one-way function, and Com an unconditionally hiding commitment scheme. With these primitives a generic separable group signature scheme can be constructed as follows.

 $\mathsf{GKG} ext{-}\mathsf{MM}$: This is the key generation algorithm SKG of the signature scheme SIG .

 $\mathsf{GKG} ext{-}\mathsf{RM}$: This is the key generation algorithm EKG of the probabilistic shadow encryption scheme SE .

GKG-GM: The group member chooses a random value x_U from the domain of f as secret key and computes his public key $y_U = f(x_U)$.

GKG-S: This algorithm chooses a hash function \mathcal{H} suitable for use in the SPK's and sets up a commitment scheme Com.

Reg: The group member sends (ID_U, y_U) to the membership manager and proves in zero-knowledge that he knows x_U such that $y_U = f(x_U)$ holds. If he is successful, the membership manager computes the signature $s_U := Sig(x_M, y_M, y_U)$ and sends it to U. The tuple (y_U, ID_U) is added to GML. The group member's output is s_U .

GSig: To sign a message $m \in \{0,1\}^*$ a group member U computes $z := Enc(y_R, y_U)$, $C := Com(y_U, r)$, where r is a random string, and the three SPK's (using informal notation)

$$S_{I} := SPK\{(\alpha, \beta) : C = Com(\alpha, \beta) \land z = Enc(y_{R}, \alpha)\}(C, z, m)$$

$$S_{II} := SPK\{(\alpha, \beta, \delta) : C = Com(\alpha, \beta) \land \alpha = f(\delta)\}(S_{I})$$

$$S_{III} := SPK\{(\alpha, \beta, \gamma) : C = Com(\alpha, \beta) \land Ver(y_{M}, \gamma, \alpha) = 1\}(S_{II}).$$

The signature on m is the tuple $(z, C, S_I, S_{II}, S_{III})$.

GVer: A group-signature $\sigma = (z, C, S_I, S_{II}, S_{III})$ on a message m can be verified by checking the SPK's S_I , S_{II} , and S_{III} .

GTrace: Given $(z, C, S_I, S_{II}, S_{III})$ and m, the revocation manager checks whether the signature is valid, shadow-decrypts z as $v = Dec(y_R, x_R, z)$, finds an (y_U, ID_U) from GML such that $(y_U, v) \in \mathcal{R}$, and computes $V := SPK\{(\alpha) : v = Dec(y_R, \alpha, z)\}(m, \sigma, v, y_U)$.

Theorem 1. In the random oracle model, the above construction is a secure group signature scheme, provided that the requirements of the used commitment scheme, shadow encryption scheme, the one-way function, and the signature scheme are satisfied.

Proof (sketch). The correctness of the signature generation is obvious.

Anonymity and unlinkability of signatures: Any two different signatures $(z, C, S_I, S_{II}, S_{III})$ and $(z', C', S'_I, S'_{II}, S'_{III})$ are computationally indistinguishable due to the security requirement for the shadow-encryption scheme, the hiding property of the commitment scheme, and the zero-knowledge property of the proofs underlying the SPK's S_I , S_{II} , and S_{III} .

Unforgeability of group signatures: Assuming the soundness of the SPK's S_I , S_{II} , and S_{III} and the commitment scheme's security, a non-group-member being able to forge signatures is also able to compute values $s_{\tilde{U}}$, $y_{\tilde{U}}$, and $x_{\tilde{U}}$ such that $y_{\tilde{U}} = f(x_{\tilde{U}})$ and $Ver(y_M, s_{\tilde{U}}, y_{\tilde{U}}) = 1$. However, the latter implies that the attacker must have existentially forged a signature of the membership manager which is assumed to be infeasible.

Unforgeability of tracing: If the revocation manager could claim that another y_U than the one whose shadow-encryption is contained in a valid signature, either the correctness property of the shadow encryption scheme or the soundness of the $SPK\ V$ would not hold.

No framing: Assume that some coalition can sign on behalf of a group member with membership key y_U . Given the unforgeability of tracing, y_U must be shadow-encrypted in z. Due to the SPK's S_I and S_{II} the coalition must be able to compute $f^{-1}(y_U)$ or to break the commitment scheme, i.e., open a commitment in two ways. Both is assumed to be infeasible.

Unavoidable traceability: Based on the same arguments as in the case of unforgeability of group signatures, we can conclude that a successfully attacking coalition must be able to forge signatures of the membership manager under a known signature-message pair attack, which is assumed to be infeasible.

This scheme indeed achieves perfect separability since the algorithms GKG-MM, GKG-RM, GKG-GM, and GKG-S can be run independent of each other. For all procedures but GSig it follows from the construction that they are efficient if the underlying primitives are efficient. To achieve efficiency for GSig as well, it seems necessary to restrict the choices of the instances of the cryptographic primitives in which case we get group signature scheme with strong separability. We refer to the next section for possible instances of the employed primitives.

It can easily be seen that the size of the group's public key and the length of signatures do not depend on the number of group members. However, in the tracing algorithm *GTrace*, the revocation manager has to check the membership key of every group member, hence the running-time of this algorithm is linear in the number of group members. We will later see that in our implementation this can be overcome and the tracing algorithm can also be made independent from the group's size. This problem could as well be solved by using semantically secure encryption (e.g., [29,37]) instead of shadow encryption. However, finding an instance of the resulting generic group signature scheme with efficient signing and verification procedures is an open problem.

We note that a (generic) identity escrow scheme can be obtained from the above scheme by replacing the SPK's in the signature generation algorithm by the underlying PK's.

5 Instances

This section provides concrete instances of cryptographic primitives that allow an efficient realization of the SPK's in the procedures GSig and GTrace of our

generic group signature scheme. All these instances are based on the discrete logarithm problem and some are additionally based on the hardness of factoring or computing roots modulo a composite. The somewhat less efficient instances which are solely based on the discrete logarithm problem are presented in the full paper.

Throughout this section, $m \in \{0,1\}^*$ denotes the message that a group member U wants to sign, k and $\epsilon > 1$ are security parameters (k denotes the bitlength of the challenges in the SPK's and ϵ controls the tightness of the zero-knowledgeness, cf. Section 3), 2^{ℓ_U} is an upper-bound on number of elements in the domain and the image of the one-way function $f: \{0,1\}^{\ell_U/2} \times \{0,1\}^{\ell_U/2} \to \{0,1\}^{\ell_U}$, and ℓ_M and ℓ_R are length-parameters of the crypto-systems chosen by the membership and the revocation manager, respectively.

5.1 A Commitment Scheme

As commitment scheme Com we apply the one due to Pedersen [39] which is information theoretically hiding and computational binding (i.e., the binding property relies on the hardness of computing discrete logarithms). Deviating from the original proposal, we use this scheme with an algebraic group of large unknown order. This does not alter the scheme's properties, but will allow us to use this group also for the efficient interval-proofs described in Section 3. An example of such a group is a subgroup of \mathbb{Z}_n^* , where n is a large RSA modulus whose factors are unknown. Either this modulus is chosen by a trusted third party or by representatives of the group members and the membership manager(s). In the latter case, the parties can employ the protocols presented in [6,24,42] to choose such a modulus jointly without the participants learning its factors. In the following we stick to the latter.

These choices have the following consequences for the affected procedures of our group signature scheme.

GKG-S (commitment part): The representatives of the group members and the membership manager jointly choose an RSA modulus $n_S > 2^{\ell_S}$, such that the factors of n_S are unknown, and a random element $h_S \in \mathbb{Z}_{n_S}^*$ (e.g., using techniques from [6,24,42]). Furthermore, they all choose a random exponent $r_i \in \{0,1\}^{\ell_S}$ and commit to $h_i = h_S^{r_i} \pmod{n_S}$ using some secure commitment scheme. If all commitments are published, they open the commitments, prove their knowledge of $\log_{h_S} h_i$, and compute $g_S = \prod_i h_i$. The parameters n_S , $G_S = \langle h_S \rangle$, g_S , h_S , and ℓ_S are published as part of the group's public key.

GSig (commitment part): A group member can commit to y_U by computing $C := g_S^{y_U} h_S^r$, where r is randomly chosen from $\{-2^{\ell_S}, \dots, 2^{\ell_S}\}$.

5.2 A Shadow Encryption Scheme

This section provides a shadow encryption scheme that is based upon the El-Gamal [22] encryption scheme which we briefly summarize. The public key of the recipient consists of a group $G_R = \langle g_R \rangle$, its prime order q_R , and $y_R = h_R^{x_R}$,

where the recipient's secret key x_R is randomly chosen from \mathbb{Z}_{q_R} . The encryption of a message $w \in G$ is a pair $(A := g_R^r, B := wy_R^r)$, where r is randomly chosen from \mathbb{Z}_{q_R} . Decryption works by computing $B/A^{x_R}(=w)$. From this scheme a shadow encryption scheme for the relation $\mathcal{R} = \{(u, v) | v = g_R^u\}$ can be derived by encrypting g_R^w instead of w, where now $w \in \mathbb{Z}_{q_R}$. Deciding whether some message $\tilde{w} \in \mathbb{Z}_{q_R}$ is indeed shadow-encrypted in a pair (A, B) can be done (when knowing the secret key) by checking whether $g_R^{\tilde{w}}$ equals B/A^{x_R} . The security properties (cf. Def. 2) of this shadow encryption scheme are inherited from the ElGamal scheme, which is equivalent to the Diffie-Hellman decision problem [46].

With this shadow encryption scheme the affected procedures of our group signature scheme are as follows. Recall that the group member has committed to y_U by $C = g_S^{y_U} h_S^r$ (cf. Section 5.1).

GKG-RM: The revocation manager chooses a group $G_R = \langle g_R \rangle$ of order q_R , a random secret key $x_R \in_R \mathbb{Z}_{q_R}$, computes $y_R := g_R^{x_R}$, and publishes (y_R, g_R, G_R, q_R) as her public key. Let $\ell_R = \lfloor \log_2 q_R \rfloor$.

Part I of GSig: In the following, we assume that $\ell_R > (\ell_U + k)\epsilon + 2$ holds. A group member U shadow-encrypts y_U by computing $A := g_R^{y_U} y_R^r$. He then can compute the first SPK as

$$\begin{split} S_I := SPK \big\{ (\alpha,\beta,\gamma): \ A &\stackrel{G_R}{=} g_R^\alpha \ \wedge \ B \stackrel{G_R}{=} g_R^\beta y_R^\alpha \ \wedge \\ C &\stackrel{G_S}{=} g_S^\beta h_S^\gamma \ \wedge \ (-2^{\ell_R} < \beta < 2^{\ell_R}) \big\} (A,B,C,m) \ . \end{split}$$

The SPK S_I shows that the value committed to by C is indeed shadow-encrypted in (A, B) under y_R .

GTrace: Knowing x_R the revocation manager can check whether some $y_{\tilde{U}}$ is shadow-encrypted in a pair (A,B) that is part of a valid group signature σ on m by testing if $g_R^{y_{U'}} \stackrel{G_R}{=} B/A^{x_R}$ holds. If it does, she can prove this by

$$V := SPK\{(\alpha): y_R \stackrel{G_R}{=} g_R^{\alpha} \wedge B/g_R^{y_{\tilde{U}}} \stackrel{G_R}{=} A^{\alpha} \}(m, \sigma, g_R^{y_{\tilde{U}}}, y_{\tilde{U}}) .$$

Remark 1. The tracing algorithm can be made independent of the number of group members if the "shadows" $g_R^{y_U}$ are stored along with y_U , ID_U , and s_U . Then, tracing can be done with a single look-up in the database. This has of course the disadvantage, that the database must be updated if the revocation manager changes her public key.

5.3 A One-Way Function $f(\cdot)$

Let the function $f: \mathtt{primes}^{\ell_U/2} \times \mathtt{primes}^{\ell_U/2} \to \{0,1\}^{\ell_U}$ be the multiplication of two $\ell_U/2$ -bit primes, i.e., $f(p'_U, p''_U) := p'_U p''_U$. For large enough ℓ_U , this function is believed to be one-way.

⁶ For a solution for the case $\ell_R < (\ell_U + k)\epsilon + 2$ we refer to the full paper.

GKG-GM: Group member U chooses two suitable primes p'_U and p''_U with $2^{\ell_U/2-1} < p'_U, p''_U < 2^{\ell_U/2}$ and computes $y_U = p'_U p''_U$. He publishes his public key y_U and keeps (p'_U, p''_U) as his secret key.

Part II of GSig: Given the factors p'_U and p''_U of y_U , where y_U is committed to by C (cf. Section 5.1), group member U can compute the third SPK as follows. He picks $r_1, r_2 \in_R \{-2^{\ell_S}, \dots, 2^{\ell_S}\}$, computes $F := g_S^{p'_U} h_S^{r_1}, L := g_S^{p''_U} h_S^{r_2}$, and

$$S_{II} := SPK \{ (\omega, \theta, \nu, \psi, \kappa) : F \stackrel{G_S}{=} g_S^{\omega} h_S^{\theta} \wedge L \stackrel{G_S}{=} g_S^{\nu} h_S^{\psi} \wedge C \stackrel{G_S}{=} F^{\nu} h_S^{\kappa} \wedge (-2^{(\ell_U/2+k)\epsilon+2} < \omega, \nu < 2^{(\ell_U/2+k)\epsilon+2}) \} (S_I, F, L) .$$

This SPK shows that the integer committed to by C is the product (in \mathbb{Z}) of the two integers committed to by F and L. Assuring that F and L are nontrivial factors of the integer committed to by C requires $(\ell_U/2+k)\epsilon+2<\ell_U$ and that the membership manager signs only y_U 's that lie between 2^{ℓ_U-1} and 2^{ℓ_U} .

5.4 A Signature Scheme

Signature schemes that are applicable must allow an efficient proof of knowledge of a membership manager's signature on y_U 's that are committed to by some $C = Com(y_U)$ (cf. Section 5.1). Typically, signature schemes require the use of a hash function as redundancy function to be existentially unforgeable. However, the need for efficient proofs requires that the redundancy function allows to compute the commitment $C' = Com(red(y_U))$ given C only. An example that allows this is $red(x) = x2^K + d$, where a randomly chosen $d \in \{0, \ldots, 2^K - 1\}$ is fixed and K is a security parameter. Note that some attacks on RSA with such simple redundancy schemes are known [27,36]. However, these attacks seem not to work, if K is chosen sufficiently large. Moreover, they are (arbitrary) chosenmessage attacks and are therefore not applicable as our construction requires only a signature scheme that is existential unforgeable under a restricted kind of chosen message attack (cf. Section 4.1). Hence, the RSA signature scheme together with this simple redundancy function seems to satisfy our requirements.

In the following we assume ℓ_U and ℓ_M are such that $\ell_U + K \leq \ell_M$ holds. Using the RSA signature scheme [43] together with this redundancy function has the following consequences.

GKG-MM: The membership manager chooses two $\ell_M/2$ -bit primes p_M' and p_M'' , computes $n_M = p_M' p_M''$, chooses a prime $e_M > 1$, selects a random integer $d \in_R \{0, \ldots, 2^K - 1\}$, publishes (n_M, e_M, d) as her public key, and stores (p_M', p_M'') as her secret key.

Reg: The group member sends (y_U, ID_U) to the membership manager and proves her (1) that y_U is the product of two primes and (2) that these primes are of size $\approx 2^{\ell_U/2}$. The first can be done with protocols from [7,10,47]; for the latter the group member computes $c_{2'} := g_S^{p'_U} h_S^{v_{p'}}$ and $c_{2''} := g_S^{p'_U} h_S^{v_{p''}}$, where

 $v_{p''}, v_{p'} \in_R \{-2^{\ell_S}, \dots, 2^{\ell_S}\}$, and carries out the protocol

$$PK^{b}\{(\alpha,\beta,\gamma,\delta,\kappa): c_{2'} \stackrel{G_{S}}{=} g_{S}^{\alpha}h_{S}^{\beta} \wedge c_{2''} \stackrel{G_{S}}{=} g_{S}^{\gamma}h_{S}^{\delta} \wedge g_{S}^{y_{U}} \stackrel{G_{S}}{=} c_{2'}^{\gamma}h_{S}^{\kappa} \wedge (-2^{\epsilon\ell_{U}/2+2} \leq \alpha, \gamma \leq 2^{\epsilon\ell_{U}/2+2})\}$$

with the membership manager. If all these proofs are fine and if y_U is an ℓ_U -bit number, the membership manager signs y_U , i.e., computes $s_U := red(y_U)^{1/e_M} \equiv (y_U 2^K + d)^{1/e_M} \pmod{n_M}$, and sends s_U to the group member who checks its validity. Finally, the two parties enter (y_U, ID_U) in the membership list GML.

Part III of GSig: We assume that $e_M = 3$ (other cases can be done similarly). Group member U computes $D := g_S^{s_U} h_S^{r_1}$, $E := g_S^{s_U^2 \pmod{n_M}} h_S^{r_2}$, with $r_1, r_2 \in_R \{-2^{\ell_S}, \dots, 2^{\ell_S}\}$, and

$$S_{III} := SPK \{ (\beta, \gamma, \psi, \lambda, \tau, \pi, \delta, \zeta, \rho, \xi) : C \stackrel{G_S}{=} g_S^{\beta} h_S^{\gamma} \wedge D \stackrel{G_S}{=} g_S^{\psi} h_S^{\lambda} \wedge E \stackrel{G_S}{=} g_S^{\tau} h_S^{\pi} \wedge E \stackrel{G_S}{=} D^{\psi} (g_S^{n_M})^{\delta} h_S^{\zeta} \wedge C^{2^K} g_S^d \stackrel{G_S}{=} E^{\psi} (g_S^{n_M})^{\rho} h_S^{\xi} \wedge (-2^{(\ell_U + k)\epsilon + 2} < \beta < 2^{(\ell_U + k)\epsilon + 2}) \wedge (-2^{(\ell_M + k)\epsilon + 2} < \psi, \tau, \delta, \rho < 2^{(\ell_M + k)\epsilon + 2}) \} (S_I, S_{II}, D, E) .$$

This SPK shows that the cubicle of the integer committed to by D equals the integer obtained when applying the redundancy function red to integer committed to by C (modulo n_M).

5.5 Efficiency Considerations and Remarks

To make the efficiency consideration easier, we first put the different parts of the signature generation algorithm together. Furthermore, we merge the SPK's S_I , S_{II} , and S_{III} into a single one.

GSig (all parts): Knowing $y_U = p'_U p''_U$, $x_U = (p'_U, p''_U)$, and s_U , group member U can sign a message $m \in \{0,1\}^*$ on the group's behalf by choosing $r_1, r_3, r_4, r_5, r_6 \in_R \{-2^{\ell_S}, \dots, 2^{\ell_S}\}$ and $r_2 \in_R \mathbb{Z}_{q_R}$ and computing $A := g_R^{r_2}$, $B := g_R^{y_U} y_R^{r_2}$, $C := g_S^{y_U} h_S^{r_1}$, $D := g_S^{s_U} h_S^{r_3}$, $E := g_S^{s_U^2} \pmod{n_M} h_S^{r_4}$, $F := g_S^{p'_U} h_S^{r_5}$, $L := g_S^{p'_U} h_S^{r_5}$, and

$$\begin{split} S_{I-III} &:= SPK \big\{ (\alpha,\beta,\gamma,\psi,\lambda,\tau,\pi,\delta,\zeta,\omega,\theta,\nu,\mu,\rho,\xi,\kappa) : \\ A &\stackrel{G_R}{=} g_R^{\alpha} \wedge B \stackrel{G_R}{=} g_R^{\beta} y_R^{\alpha} \wedge C \stackrel{G_S}{=} g_S^{\beta} h_S^{\gamma} \wedge D \stackrel{G_S}{=} g_S^{\psi} h_S^{\lambda} \wedge \\ E &\stackrel{G_S}{=} g_S^{\tau} h_S^{\pi} \wedge E \stackrel{G_S}{=} D^{\psi} (g_S^{n_M})^{\delta} h_S^{\zeta} \wedge F \stackrel{G_S}{=} g_S^{\omega} h_S^{\theta} \wedge \\ L &\stackrel{G_S}{=} g_S^{\nu} h_S^{\mu} \wedge C^{2^K} g_S^{d} \stackrel{G_S}{=} E^{\psi} (g_S^{n_M})^{\rho} h_S^{\xi} \wedge C \stackrel{G_S}{=} F^{\nu} h_S^{\kappa} \wedge \\ (-2^{\epsilon(\ell_U + k) + 2} < \beta < 2^{\epsilon(\ell_U + k) + 2}) \wedge (-2^{\epsilon(\ell_M + k) + 2} < \psi, \tau, \delta, \rho < 2^{\epsilon(\ell_M + k) + 2}) \wedge \\ (-2^{\epsilon(\ell_U / 2 + k) + 2} < \omega, \nu < 2^{\epsilon(\ell_U / 2 + k) + 2}) \big\} (A, B, C, D, E, F, L, m) . \end{split}$$

Provided that elements from G_S and G_R have roughly the same size, the signer's computational load is about 17 multi-exponentiations and the verifier's computational effort is about 10 multi-exponentiations. If elements in G_R and G_S are about ℓ_R and ℓ_S bits long, respectively, a signature takes $5\ell_S + 2\ell_R + \ell_R + 2(\epsilon(\ell_U/2+k)+2) + (\epsilon(\ell_U+k)+2) + 8(\epsilon(\ell_S+k)+2) + 4(\epsilon(\ell_M+k)+2) + k$ bits. The following choices of security parameter (such that $\ell_R > \epsilon(\ell_U+k) + 2$ (cf. Section 5.2), $(\ell_U/2+k)\epsilon + 2 < \ell_U$ (cf. Section 5.3), and $\ell_M > \ell_U + K$ (cf. Section 5.4)) could be used: $\epsilon = 9/8$, $\ell_U = 768$, $\ell_R = 1100$, $\ell_M = 1630$, $\ell_S = 1024$, K = 850, and k = 160. With these choices a signature is about 3, 5 kilobytes long. Compared to the most efficient non-separable group signature schemes [9,11,12], we lose roughly a factor of 3 in terms of the length of signatures as well as the number of exponentiations.

6 Extensions and Open Problems

In order to keep the system manageable it is desirable to be able to remove group members from the group. This implies that the group's public key must be changed each time that a group member is removed and also that each group signature is time-stamped. In principle, group members can be removed by changing the membership manager's key and by issuing new certificates to the remaining group members. A more elegant way to realize this is using so-called *one-way accumulators* (OWA) [3,5,19], in particular by using the efficient realization of an OWA given in [3]. Details are provided in the full paper.

Another possible extension is to distribute the role the membership and the revocation manager among several parties while still ensuring the separability with respect to all parties. This can be done by combining standard secret sharing techniques with the proof techniques described in Section 3, in particular those in Section 3.4.

Further research is required for an exact security analysis of the signature scheme presented in Section 5.4. Finding a signature schemes with simple redundancy functions and designing a separable group signature scheme with efficient signature generation and verification that is exclusively based on standard assumptions are challenging open problems.

Acknowledgments

The authors are grateful to Ivan Damgård and Louis Salvail for discussions.

References

- G. Ateniese and G. Tsudik. Group signatures à la carte. In ACM Symposium on Discrete Algorithms, 1999.
- G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In Proc. of Financial Cryptography '99, 1999.

- 3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 480–494.
- M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In First ACM CCS, pp. 62–73. ACM, 1993.
- J. Benaloh and M. de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Sinatures. In Advances in Cryptology — EUROCRYPT '93, vol. 765 of LNCS, pp. 274-285.
- D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In Advances in Cryptology — CRYPTO '97, vol. 1296 of LNCS, pp. 425–439.
- J. Boyar, K. Friedl, and C. Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. *Journal of Cryptology*, 4(3):185–206, 1991.
- J. Camenisch. Efficient and generalized group signatures. In Advances in Cryptology EUROCRYPT '97, vol. 1233 of LNCS, pp. 465–479.
- J. Camenisch and M. Michels. A group signature scheme based on an RSA-variant. Tech. Rep. RS-98-27, BRICS, Dept. of Comp. Sci., University of Aarhus, preliminary version in Advances in Cryptology — ASIACRYPT '98, vol. 1514 of LNCS.
- J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In Advances in Cryptology — EUROCRYPT '99, vol. 1592 of LNCS, pp. 107–122.
- 11. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology CRYPTO '97*, vol. 1296 of *LNCS*, pp. 410–424.
- 12. J. L. Camenisch. Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich, 1998.
- A. Chan, Y. Frankel, and Y. Tsiounis. Easy come easy go divisible cash. GTE Technical Report, preliminary version appeared in Advances in Cryptology — EU-ROCRYPT '98, vol. 1403 of LNCS, pp. 561–575.
- D. Chaum. Zero-knowledge undeniable signatures. In Advances in Cryptology EUROCRYPT '90, vol. 473 of LNCS, pp. 458–464.
- D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In Advances in Cryptology EUROCRYPT '87, vol. 304 of LNCS, pp. 127–141.
- D. Chaum and T. P. Pedersen. Wallet databases with observers. In Advances in Cryptology — CRYPTO '92, vol. 740 of LNCS, pp. 89–105.
- 17. D. Chaum and E. van Heyst. Group signatures. In Advances in Cryptology EUROCRYPT '91, vol. 547 of LNCS, pp. 257–265.
- 18. L. Chen and T. P. Pedersen. New group signature schemes. In *Advances in Cryptology EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 171–181.
- 19. R. Cramer. Personal communication.
- R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Advances in Cryptology CRYPTO '94, volume 839 of LNCS, pp. 174–187. Springer Verlag, 1994.
- I. B. Damgård. Practical and provable secure release of a secret and exchange of signatures. In Advances in Cryptology — EUROCRYPT '93, vol. 765 of LNCS, pp. 200–217.
- T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In Advances in Cryptology — CRYPTO '84, vol. 196 of LNCS, pp. 10–18.
- A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In Advances in Cryptology — CRYPTO '86, vol. 263 of LNCS, pp. 186–194.

- Y. Frankel, P. D. MacKenzie, and M. Yung. Robust efficient distributed RSA-key generation. In STOC'98, pp. 663-672, 1998.
- E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Advances in Cryptology — CRYPTO '97, vol. 1294 of LNCS, pp. 16–30.
- R. Gennaro, H. Krawczyk, and T. Rabin. RSA-based undeniable signatures. In Advances in Cryptology — CRYPTO '97, vol. 1296 of LNCS, pp. 132–149.
- 27. M. Girault and J.-F. Misarsky. Selective forgery of RSA using redundancy In *Advances in Cryptology EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 495-507.
- 28. S. Goldwasser and M. Bellare. Lecture notes on cryptography, June 1997.
- 29. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, Apr. 1984.
- 30. J. Kilian and E. Petrank. Identity escrow. In Advances in Cryptology CRYPTO '98, vol. 1642 of LNCS, pp. 169–185.
- 31. S. J. Kim, S. J. Park, and D. H. Won. Convertible group signatures. In *Advances in Cryptology ASIACRYPT '96*, vol. 1163 of *LNCS*, pp. 311–321.
- 32. W.-B. Lee and C.-C. Chang. Efficient group signature scheme based on the discrete logarithm. *IEE Proc. Comput. Digit. Tech.*, 145(1):15–18, 1998.
- 33. C. H. Lim and P. J. Lee. On the security of convertible group signatures. *Electronics Letters*, 1996.
- M. Michels. Comments on some group signature schemes. Technical Report TR-96-3-D, Dept. of Comp. Sci., Univ. of Technology, Chemnitz-Zwickau, Nov. 1996.
- 35. M. Michels and M. Stadler. Generic constructions for secure and efficient confirmer signature schemes. In *Advances in Cryptology EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 406–421.
- 36. J.-F. Misarsky. A multiplicative attack using LLL Algorithm on RSA signatures with redundancy. In *Advances in Cryptology CRYPTO '97*, vol. 1294 of *LNCS*, pp. 221-234.
- 37. S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, April 1988.
- 38. S. J. Park, I. S. Lee, and D. H. Won. A practical group signature. In *Proc. of the* 1995 Japan-Korea Workshop on Information Security and Cryptography, 1995.
- T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Advances in Cryptology – CRYPTO '91, vol. 576 of LNCS, pp. 129–140.
- 40. H. Petersen. How to convert any digital signature scheme into a group signature scheme. In *Security Protocols Workshop*, vol. 1361 of *LNCS*, pp. 177 190, 1997.
- 41. D. Pointcheval. Les Preuves de Connaissance et leurs Preuves de Sécurité. PhD thesis, Université de Caen, 1996.
- 42. G. Poupard and J. Stern. Generation of shared RSA keys by two parties. In Advances in Cryptology ASIACRYPT '98, vol. 1514 of LNCS, pp. 11-24.
- 43. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120–126, 1978.
- 44. A. de Santis, G. di Crescenzo, G. Persiano, and M. Yung. On Monotone Formula Closure of SZK. 35th FOCS, IEEE, pp. 454–465, 1994.
- 45. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
- Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In International Workshop on Practice and Theory in Public Key Cryptography, 1998.
- 47. J. van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. In *Advances in Cryptology CRYPTO '87*, vol. 293 of *LNCS*, pp. 128–134.

A Forward-Secure Digital Signature Scheme

Mihir Bellare and Sara K. Miner

Dept. of Computer Science, & Engineering
University of California at San Diego, 9500 Gilman Drive
La Jolla, CA 92093, USA
{mihir,sminer}@cs.ucsd.edu

URL: http://www-cse.ucsd.edu/users/{mihir,sminer}

Abstract. We describe a digital signature scheme in which the public key is fixed but the secret signing key is updated at regular intervals so as to provide a *forward security* property: compromise of the current secret key does not enable an adversary to forge signatures pertaining to the past. This can be useful to mitigate the damage caused by key exposure without requiring distribution of keys. Our construction uses ideas from the Fiat-Shamir and Ong-Schnorr identification and signature schemes, and is proven to be forward secure based on the hardness of factoring, in the random oracle model. The construction is also quite efficient.

1 Introduction

This paper presents a digital signature scheme having a novel security property, namely "forward security." This is a means to mitigate the damage caused by key exposure.

1.1 The Key Exposure Problem

In practice the greatest threat against the security of a digital signature scheme is exposure of the secret (signing) key, due to compromise of the security of the underlying system or machine storing the key. The danger of successful cryptanalysis of the signature scheme itself is hardly as great as the danger of key exposure, as long as we stick to well-known schemes and use large security parameters.

The most widely considered solution to the problem of key exposure is distribution of the key across multiple servers via secret sharing [18,5]. There are numerous instantiations of this idea including threshold signatures [7] and proactive signatures [14]. Distribution however is quite costly. While a large corporation or a certification authority might be able to distribute their keys, the average user, with just one machine, does not have this option. Thus while we expect digital signatures to be very widely used, we do not expect most people to have the luxury of splitting their keys across several machines. Furthermore even when possible, distribution may not provide as much security as one might imagine. For example, distribution is susceptible to "common-mode failures:" a

system "hole" that permits break-in might be present on all the machines since they are probably running a common operating system, and once found, all the machines can be compromised.

Other ways of protecting against key exposure include use of protected hardware or smartcards, but these can be costly or impractical.

1.2 Forward Secure Signatures

The goal of forward security is to protect some aspects of signature security against the risk of exposure of the secret signing key, but in a simple way, in particular without requiring distribution or protected storage devices, and without increasing key management costs.

How is it possible to preserve any security in the face of key exposure without distribution or protected devices? Obviously, we cannot hope for total security. Once a signing key is exposed, the attacker can forge signatures. The idea of "forward security" is however that a distinction can be made between the security of documents pertaining to (meaning dated in) the past (the time prior to key exposure) and those pertaining to the period after key exposure.

The Key evolution paradigm. A user begins, as usual, by registering a public key PK and keeping private the corresponding secret key, which we denote SK_0 . The time during which the public key PK is desired to be valid is divided into periods, say T of them, numbered $1, \ldots, T$. While the public key stays fixed, the user "evolves" the secret key with time. Thus in each period, the user produces signatures using a different signing key: SK_1 in period 1, SK_2 in period 2, and so on. The secret key in period i is derived as a function of the one in the previous period; specifically, when period i begins, the user applies a public oneway function i to i to get i to get i that point, the user also deletes i because the previous keys i because the latter was a oneway function of the previous key.) The key evolution paradigm is illustrated in Figure 1.

A signature always includes the value j of the time period during which it was produced, so that it is viewed as a pair $\langle j, \zeta \rangle$. The verification algorithm takes the (fixed) public key PK, a message and candidate signature, and verifies that the signature is valid in the sense that it was produced by the legitimate user in the period indicated in the signature. We stress that although the user's secret key evolves with time, the public key stays fixed throughout, so that the signature verification process is unchanged, as are the public key certification and management processes.

The number of periods and the length of each period are parameters of choice. For example, we might want to use the scheme under a certain public key for one year, with daily updates, in which case T=365 and each period has length one day.

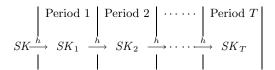


Fig. 1. Paradigm for a forward secure signature scheme: Secret key in a given period is a one-way function h of the secret key in the previous period.

SECURITY BENEFITS. The security benefit of this key evolution paradigm is that loss of the current secret key will not enable the adversary to forge signatures with a "date" prior to the one at which key exposure occurred. This "date" has a precise meaning: it is the value of the period during which the signature is (claimed to be) produced, which as indicated above is always included in the signature itself. This can be viewed as a means of protecting the authenticity of past transactions even in the event of key exposure. It is quite a useful property and can mitigate the damage of key exposure in several ways.

The following example, based on an idea due to Diffie, illustrates. Suppose Eve's mortgage payment is due on the first of each month. When it is delivered to the bank, the bank issues a signed and dated receipt. First suppose we are using a standard (not forward-secure) signature scheme. Then for example, if Eve paid \$1,000 on January 1st 1999, the text "Eve paid \$1,000 on January 1st, 1999" is signed under the public key of the bank. On February 1st, Eve is broke and doesn't pay, and of course the bank issues no receipt. But on February 2nd, Eve cracks the bank computer. Under this normal signing scheme, she gets the signing key, and can forge a note saying "Eve paid \$1,000 on February 1st, 1999". But not under a forward secure scheme. Under a forward secure scheme (assuming daily key updates) the signature that would have been produced on February 1st would have the form $\langle 32, \zeta \rangle$ where ζ is a tag for the text "Eve paid \$1,000," and we are assuming for simplicity that the scheme is initialized on January 1st, so that February 1st, being the 32nd day of the year, has corresponding date or period the number "32". Eve gets only the signing key for February 2nd, the 33rd period. She could forge any signature of the form $\langle j,\zeta\rangle$ for $j\geq 33$, for any text of her choice, but not one of the form $\langle 32, \zeta \rangle$. So she cannot claim to have paid up on February 1st.

RELATION TO TIME-STAMPING. Time-stamping signed documents via a trusted time stamping authority (cf. [13]) can also provide a similar kind of security, but this requires that one make use of such an authority, which is costly in various ways. Forward security may be viewed as providing a certain kind of time-stamp, namely one that is secure against forgery by an adversary who obtains the current secret key. (However it assumes the signer is honest since the signer could of course "forge" time-stamps if it wanted by simply not deleting previous keys.)

HISTORY. The term (perfect) "forward secrecy" was first used in [12] in the context of session key exchange protocols, and later in [8]. The basic idea, as described in [8], is that compromise of long-term keys does not compromise past session keys, meaning that past actions are protected in some way against loss of the current key, the same basic idea as here in a different context. The above paradigm and the idea of a digital scheme with forward security were suggested by Ross Andersen in an invited lecture at the ACM CCS conference [1]. He left as an open problem to find such a scheme. In this paper we provide the first solution. We also provide a formal adversarial model and notion of security with respect to which the security of schemes can be assessed. In particular we suggest the inclusion of the current date in the signature and the use of the claimed date as the determinant of whether something is past or present from the forward security point of view.

1.3 Construction of a Forward Secure Digital Signature Scheme

FIRST IDEAS. It is trivial to design a forward secure signature scheme if you allow any of scheme parameters to grow proportionally with the number T of time periods over which the scheme is supposed to be valid; specifically, if the size of the public key, secret key, or the signature itself is allowed to be proportional to T. We briefly describe these methods in Section 2. However, any such method is impractical.

The first reasonable solution, also described in Section 2, involves the use of binary tree based certification chains, and results in secret keys and signatures of size linear in $\lg(T)$. Ideally however we would like a solution in which the size of the public key, secret key, and signature does not depend on the number of time periods. But achieving forward security in this setting does not seem so easy. A natural starting idea is to try to find a suitable key evolution mechanism for standard RSA or El Gamal type schemes, but we were unable to find a secure one in either case.

OUR SCHEME. To get a forward-secure digital signature scheme of the desired kind, we turn to a different paradigm: construction of signature schemes based on identification schemes. We present a forward secure digital signature scheme based on ideas underlying the Fiat-Shamir [10] and Ong-Schnorr [16] identification and signature schemes. The feature of these schemes that is crucial to enable secure key evolution is that even the signer does not need to know the factorization of the modulus on which the scheme is based, but just the square roots of some public values. We evolve the secret key via squaring, which is a one-way function in this setting.

NOTION OF SECURITY. To provide assurance that our scheme has the forward security property, we first provide a formal definition of forward security for digital signatures, extending the notion of security for standard digital signatures from [11] to allow for key exposure attacks. Our model allows the adversary to mount a chosen-message attack, then expose the signing key at some current period j of its choice. It is successful if it can forge a signature of the form $\langle i, \zeta \rangle$

for some message M where i < j pertains to a period prior to that of the key exposure. The scheme is secure if this task is computationally infeasible.

FORWARD SECURITY OF OUR SCHEME. We then show that our scheme meets this notion of forward security assuming it is hard to factor Blum-Williams integers. This proof is in the random oracle model [3], meaning it assumes that a certain hash function used in the scheme has random behavior.

Our security analysis proceeds in two steps. We first define a notion of a "forward secure identification (ID) scheme," design such a scheme, and prove it secure based on the hardness of factoring Blum-Williams integers. Our signature scheme is obtained from the ID scheme by the usual transformation of ID schemes into signature schemes (in which the challenge is specified as a hash of the message and the commitment). We then show that this transformation preserves forward security. (The transformation is known to preserve security in the standard sense [17,15], but here we are considering a new security feature.)

We stress one issue with regard to forward secure ID schemes: they are artificial constructs, in the sense that the security notion we put forth there, although mathematically viable, does not correspond to any real attack in a practical setting. (This is because identification is an on-line activity, unlike signature verification; one cannot return to the past and try to identify oneself.) But this does not matter in our setting, because for us, the ID scheme is simply a convenient abstraction that enables our construction of a forward secure signature scheme, and the latter is a real and useful object.

Our goal here has been to present the simplest possible scheme that has the forward security property, is reasonably efficient, and can be proven secure. Improvements and alternatives seem possible. In particular one could try to build a scheme only in the Ong-Schnorr style (rather than a combination of that with Fiat-Shamir as we do). This will reduce key sizes, but increase computation time, and the analysis (one could try to extend [19]) seems more involved.

A NOTE ON SYNCHRONIZATION. One might imagine that a scheme using time periods in the way we do will impose a requirement for clock synchronization, arising from disagreements near the time period boundaries over what is truly the current time. However, in our signature scheme, discrepancies over time periods do not cause problems. This is due to the fact that the time period in which the message was signed is stamped on the message, and the verifier looks at this stamp to determine which verification process to use. That is, it doesn't matter what the time period of verification is; the signature is verified using the information from the time period when the signature actually occurred.

2 Some Simple Solutions

We summarize several simple ways to design a forward secure signature scheme. The first three methods result in impractical schemes: some parameter (the public key size, the secret key size, or the signature size) grows linearly with the number T of time periods over which the public key is supposed to be valid. The

methods are nonetheless worth a brief look in order to better understand the problem and to lead into the fourth method. The latter, which we call the tree scheme, is a reasonable binary certification tree based solution in which key and signature sizes are logarithmic in T. Nonetheless it would be preferable if even this dependency is avoided, which is accomplished by our main scheme presented in Section 3.

In the following we make use of some fixed standard digital signature scheme whose signing and verifying algorithms are denoted SGN and VF respectively.

LONG PUBLIC AND SECRET KEYS. The signer generates T pairs $(p_1, s_1), \ldots, (p_T, s_T)$ of matching public and secret keys for the standard scheme. He sets the public key of the key evolving scheme to (p_1, \ldots, p_T) and his starting (base) secret key for the key evolving scheme to (s_0, s_1, \ldots, s_T) where s_0 is the empty string. Upon entering period j the signer deletes s_{j-1} , so that he is left with key (s_j, \ldots, s_T) . The signature of a message m in period j is $\langle j, \operatorname{SGN}_{s_j}(m) \rangle$. A signature $\langle j, \zeta \rangle$ on a message m is verified by checking that $\operatorname{VF}_{p_j}(m, \zeta) = 1$. This method clearly provides forward security, but the size of the keys (both public and secret) of the key evolving scheme depends on T, which is not desirable.

LONG SECRET KEY ONLY. Andersen [1] suggested the following variant which results in a short public key but still has a secret key of size proportional to T. Generate T key-pairs as above, and an additional pair (p, s). Let $\sigma_j = \operatorname{SGN}_s(j||p_j)$ be a signature (with respect to p) of the value j together with the j-th public key, for $j=1,\ldots,T$. This done, delete s. The public key of the key evolving scheme is p, and the signer's starting (base) secret key for the key evolving scheme is $(s_0,\sigma_0,s_1,\sigma_1,\ldots,s_T,\sigma_T)$ where s_0,σ_0 are set to the empty string. Upon entering period p the signer deletes p and the is left with secret key p and p and p and the signature of a message p in period p is p and p are set to the empty string. Calculate that p are set to the empty string. Upon entering period p and p are set to the empty string. Upon entering period p and p are set to the empty string. Upon entering period p and p are set to the empty string. Upon entering period p and p are set to the empty string. Upon entering period p are set to the empty string. Upon entering period p and p are set to the empty string. Upon entering period p and p are set to the empty string. Upon entering period p and p are set to the empty string. Upon entering period p are set to the empty string. Upon entering period p are set to the empty string. Upon entering period p are set to the empty string. Upon entering period p are set to the empty string. Upon entering period p are set to the empty string. Upon entering period p are set to the empty string.

LONG SIGNATURES. The size of both the public and the secret key can be kept small by using certification chains, but this results in large signatures. The signer begins by generating a key pair (p_0, s_0) of the standard scheme. He sets the public key of the key evolving scheme to p_0 and the starting (base) secret key of the key evolving scheme to s_0 . At each period a new key is generated and certified with respect to the previous key, which is then deleted. The certificate chain is included in the signature. To illustrate, at the start of period 1 the signer creates a new key pair (p_1, s_1) , sets $\sigma_1 = \text{SGN}_{s_0}(1||p_1)$, and deletes s_0 . The signature of a message m in period 1 is $\langle 1, (\text{SGN}_{s_1}(m), p_1, \sigma_1) \rangle$. A signature $\langle 1, (\alpha, q_1, \tau_1) \rangle$ on a message m is verified by checking that $\text{VF}_{q_1}(m, \alpha) = 1$ and $\text{VF}_{p_0}(1||q_1, \tau_1) = 1$. This continues iteratively, so that at the start of period $j \geq 2$ the signer, in possession of $p_1, \sigma_1, \ldots, p_{j-1}, \sigma_{j-1}$ and the secret key s_{j-1} of the previous period,

creates a new key pair (p_j, s_j) , sets $\sigma_j = \text{SGN}_{s_{j-1}}(j || p_j)$, and deletes s_{j-1} . The signature of a message m in period j is $\langle j, (\text{SGN}_{s_j}(m), p_1, \sigma_1, \ldots, p_j, \sigma_j) \rangle$. A signature $\langle j, (\alpha, q_1, \tau_1, \ldots, q_j, \tau_j) \rangle$ on a message m is verified by checking that $\text{VF}_{q_j}(m, \alpha) = 1$ and $\text{VF}_{q_{i-1}}(i || q_i, \tau_i) = 1$ for $i = 2, \ldots, j$ and $\text{VF}_{p_0}(1 || q_1, \tau_1) = 1$. Again, forward security is clearly provided. Furthermore both the public and the secret key are of size independent of T. But the size of a signature grows linearly with T. Also note that although the size of the signer's secret key has shrunk, the signer does have to store the list of public keys and their tags, which means it must use storage linear in T, which is not desirable.

BINARY CERTIFICATION TREE SCHEME. Andersen's scheme above can be viewed as building a certification tree of depth 1 and arity T, while the "long signature" solution just presented builds a standard certification chain, which is a tree of depth T and arity 1. Signature size is linear in the depth, and key size is linear in the arity. It is natural to form a "hybrid" scheme so as to get the best possible tradeoff between these sizes. Namely we use a binary tree with T leaves. Each key certifies its two children keys. The leaf keys are the actual message signing keys, one for each period, and the root public key is the public key of the key evolving scheme. The tree grows down from the root in a specific manner, with nodes being created dynamically. A node certifies its two children, and as soon as the two children nodes are created and certified, the secret key corresponding to the parent node is deleted. Appropriately done this results in a forward secure scheme with secret key size and signature size linear in $\lg(T)$, and fixed public key size. Furthermore the total amount of information (whether secret or not) stored at any time by the signer, and the computation time for key updates, are linear in $\lg(T)$.

One has to be a little careful to build the tree in the right way and delete keys at the right times, so let us give a few more details. For simplicity assume $T=2^{\tau}$ is a power of two. Imagine a binary tree with T leaves. The root is at level 0 and the leaves are at level τ . The root is labeled with the empty string ε , and if a node has label the binary string w then its left child is labeled w0 and its right child is labeled w1. Associate to a node with label w a key pair (p[w], s[w]). (The entire tree never exists at any time, but is useful to imagine.) The public key of the key evolving scheme is $p[\varepsilon]$. Let $\langle i, n \rangle$ denote the binary representation of the integer i-1 as a string of exactly n bits.

In period $j \geq 1$ the signer signs data under $s[\langle j, \tau \rangle]$, and attaches a certification chain based on the path from leaf $\langle j, \tau \rangle$ to the root of the tree. During period j, a certain subtree T of the full tree exists. It consists of all nodes on the path from leaf $\langle j, \tau \rangle$ to the root, plus the right sibling of any of these nodes that is a left child of its parent. All childless nodes w in the subtree that are right children of their parents have their secret key s[w] still attached; for all other nodes, it has been deleted. Notice that if a node w has s[w] still "alive," its descendent leaves have names $\langle i, \tau \rangle$ with i > j, so correspond to future periods. When the signer enters period j+1 (here j < T) it must update its subtree. This involves (possibly) creation of new nodes, and deletion of old ones in such a way that the property of the subtree described above is preserved. The signer

moves up from $\langle j,\tau\rangle$ (deleting this leaf node and its keys) and stops at the first node whose left (rather than right) child is on the path from $\langle j,\tau\rangle$ to the root. Call this node w. We know that the secret key of its right child w1 is alive. If w1 is a leaf, the update is complete. Else, create its children by picking new key pairs for each child, and delete the secret key at w. Then move left, and continue this process until a leaf is reached.

One can show that the scheme has the desired properties.

3 Our Forward-Secure Signature Scheme

KEYS AND KEY GENERATION. The signer's public key contains a modulus N and l points U_1, \ldots, U_l in Z_N^* . The corresponding base secret key SK_0 contains points S_1, \ldots, S_l in Z_N^* , where S_j is a 2^{T+1} -th root of U_j for $j = 1, \ldots, T$. The signer generates the keys by running the following key generation process, which takes as input the security parameter k determining the size of N, the number l of points in the keys, and the number T of time periods over which the scheme is to operate.

```
Algorithm KG(k,l,T)
Pick random, distinct k/2 bit primes p,q, each congruent to 3 \mod 4
N \leftarrow pq
For i=1,\ldots,l do S_i \overset{R}{\leftarrow} Z_N^*; U_i \leftarrow S_i^{2^{(T+1)}} \mod N EndFor SK_0 \leftarrow (N,T,0,S_{1,0},\ldots,S_{l,0}); PK \leftarrow (N,T,U_1,\ldots,U_l)
Return (PK,SK_0)
```

As the code indicates, the keys contain some sundry information in addition to that mentioned above. Specifically the number T of time periods is thrown into the public key. This enables the verifier to know its value, which might vary with different signers. It is also thrown into the secret key for convenience, as is the modulus N. The third component of the base secret key, namely "0", is there simply to indicate that this is the base key, in light of the fact that the key will be evolving later. The modulus is a Blum-Williams integer, meaning the product of two distinct primes each congruent to 3 mod 4. We refer to U_i as the i-th component of the public key.

The public key PK is treated like that of any ordinary signature scheme as far as registration, certification and key generation are concerenced. The base secret key SK_0 is stored privately. The factors p,q of N are deleted once the key generation process is complete, so that they are not available to an attacker that might later break into the system on which the secret key is stored.

KEY EVOLUTION. During time period j, the signer signs using key SK_j . This key is generated at the start of period j, by applying a key update algorithm to the key SK_{j-1} . (The latter is the base secret key when j=1.) The update algorithm is presented below. It squares the l points of the secret key at the previous stage to get the secret key at the next stage.

Algorithm
$$Upd(SK_{j-1},j)$$
 where $SK_{j-1}=(N,T,j-1,S_{1,j-1},\ldots,S_{l,j-1})$ For $i=1,\ldots,l$ do $S_{i,j}\leftarrow S_{i,j-1}^2 \bmod N$ EndFor $SK_j\leftarrow (N,T,j,S_{1,j},\ldots,S_{l,j})$ Return SK_j

Once this update is performed, the signer deletes the key SK_{j-1} . Since squaring modulo N is a one-way function when the factorization of N is unknown, it is computationally infeasible to recover SK_{j-1} from SK_j . Thus key exposure during period j will yield to an attacker the current key (and also future keys) but not past keys. We refer to $S_{i,j}$ as the i-th component of the period j secret key. Notice that the components of the secret key for period j are related to those of the base key as follows:

$$(S_{1,j},\ldots,S_{l,j}) = (S_{1,0}^{2^j},\ldots,S_{l,0}^{2^j}).$$
 (1)

We will use this later.

The length of a time period (during which a specific key is in use) is assumed to be globally known. The choice depends on the application and desired level of protection against key exposure; it could vary from seconds to days.

Significant Signature generation during period j is done via the following algorithm, which takes as input the secret key SK_j of the current period, the message M to be signed, and the value j of the period itself, to return a signature $\langle j, (Y, Z) \rangle$, where Y, Z are values in Z_N^* . The signer first generates the "committeent" Y, and then hashes Y and the message M via a public hash function $H: \{0,1\}^* \to \{0,1\}^l$ to get an l-bit "challenge" $c = c_1 \dots c_l$ which is viewed as selecting a subset of the components of the public key. The signer then computes a 2^{T+1-j} -th root of the product of the public key components in the selected subset, and multiplies this by a 2^{T+1-j} -th root of Y to get the value Z. This is detailed in the signing algorithm below.

$$\begin{aligned} \text{Algorithm } Sgn^H_{SK_j}(M,j) \text{ where } SK_j &= (N,T,j,S_{1,j},...S_{l,j}) \\ R &\stackrel{R}{\leftarrow} Z_N^* \; ; \; Y \leftarrow R^{2^{(T+1-j)}} \; \text{mod } N \; ; \; c_1 \ldots c_l \leftarrow H(j,Y,M) \\ Z \leftarrow R \cdot \prod_{i=1}^l S_{i,j}^{c_i} \; \text{mod } N \\ \text{Output } \langle j, (Y,Z) \rangle \end{aligned}$$

Thus, in the first time period, the signer is computing 2^T -th roots; in the second time period, 2^{T-1} -th roots; and so on, until the last time period, where it is simply computing square roots. Notice that in the last time period, we simply have the Fiat-Shamir signature scheme. (The components of the secret key SK_T are square roots of the corresponding components of the public key.) The hash function $H: \{0,1\}^* \to \{0,1\}^l$ will be assumed in the security analysis to be a random oracle.

Notice that the index j of the period during which the signature was generated is part of the signature itself. This provides some sort of "time-stamp", or claimed time-stamp, and is a crucial element of the scheme and model, since security will pertain to the ability to generate signatures having such "time-stamps" with value earlier than the current date.

VERIFYING. Verification of a candidate signature $\langle j, (Y, Z) \rangle$ for a given message M with respect to a given public key PK is performed via the following process:

Algorithm
$$Vf_{PK}^H(M,\langle j,(Y,Z)\rangle)$$
 where $PK=(N,T,U_1,...,U_l)$ $c_1\ldots c_l \leftarrow H(j,Y,M)$ If $Z^{2^{(T+1-j)}}=Y\cdot\prod_{i=1}^l U_i^{c_i} \bmod N$ Then return 1 Else return 0

Note the verification process depends on the claimed "time-stamp" or period indicator j in the signature, meaning that the period j too is authenticated.

The signature scheme is fully specified by the above four algorithms. We let $\mathsf{FSIG}[k,l,T]$ denote our scheme when the parameters are fixed as indicated.

COST. The scheme FSIG is quite efficient. Signing in period j takes T+1-j+l/2 multiplications modulo N on the average. (So the worst case cost does not exceed T+1+l multiplications, and in fact the scheme gets faster as time progresses.) For typical values of the parameters, this can be less than the cost of a single exponentiation, making signing cheaper than in RSA based schemes. Verification has the same cost as signing. (We ignore the cost of hashing, which is lower than that of the modular arithmatic.)

Like the Fiat-Shamir scheme, however, the key sizes are relatively large, being proportional to l. The size of the public key can be reduced by having its components U_1, \ldots, U_l specified implicitly rather than explicitly, as values of a random-oracle hash function on some fixed points. That is, the signer can choose some random constant U, say 128 bits long, and then specify small values a_1, \ldots, a_l such that $H^*(U, a_i)$ is a square modulo N, where H^* is a hash function with range Z_N^* . The public key is now $(N, T, U, a_1, \ldots, a_l)$. Since N is a Blum-Williams integer, the signer can then compute a 2^{T+1} -th root of $u_i = H(U, a_i)$, and thereby have a secret key relating to the public key in the same way as before. The average size of the list a_1, \ldots, a_l is very small, about $O(l \lg(l))$ bits. Unfortunately it is not possible to similarly shrink the size of the secret key in this scheme, but moving to post-Fiat-Shamir identification-derived signature schemes, one can get shorter keys.

VALIDITY OF GENUINE SIGNATURES. Before we discuss security, we should check that signatures generated by the signing process will always be accepted by the verification process.

Proposition 1. Let $PK = (N, T, U_1, \ldots, U_l)$ and $SK_0 = (N, T, 0, S_{1,0}, \ldots, S_{l,0})$ be a key pair generated by the above key generation algorithm. Let $\langle j, (Y, Z) \rangle$ be an output of $Sgn_{PK}^H(M, j)$. Then $Vf_{PK}(M, \langle j, (Y, Z) \rangle) = 1$.

Proof: Let $c_1 cdots c_l \leftarrow H(j, Y, M)$. We check that $Z^{2^{(T+1-j)}} = Y \cdot \prod_{i=1}^l U_i^{c_i} \mod N$ using Equation (1) and the fact that the signature was correctly generated:

$$\begin{split} Z^{2^{(T+1-j)}} &= \left(R \cdot \prod_{i=1}^{l} S_{i,j}^{c_i}\right)^{2^{(T+1-j)}} \bmod N \\ &= R^{2^{(T+1-j)}} \cdot \prod_{i=1}^{l} S_{i,0}^{c_i \cdot 2^{j+(T+1-j)}} \bmod N \end{split}$$

$$\begin{split} &= Y \cdot \prod_{i=1}^{l} S_{i,0}^{c_i \cdot 2^{(T+1)}} \mod N \\ &= Y \cdot \prod_{i=1}^{l} U_i^{c_i} \mod N \;, \end{split}$$

as desired.

SECURITY MODEL. We wish to assess the forward security of our scheme. To do this effectively we must first pin down an appropriate model; what can the adversary do, and when is it declared successful?

Recall the goal is that even under exposure of the current secret key it should be computationally infeasible for an adversary to forge a signature "with respect to" a previous secret key. The possibility of key exposure is modeled by allowing the adversary a "break-in" move, during which it can obtain the secret key SK_j of the current period j. The adversary is then considered successful if it can create a valid forgery where the signature has the form $\langle i, (Y, M) \rangle$ for some i < j, meaning is dated for a previous time period. The model is further augmented to allow the adversary a chosen-message attack prior to its break-in. In that phase, the adversary gets to obtain genuine signatures of messages of its choice, under the keys SK_1, SK_2, \ldots in order, modeling the creation of genuine signatures under the key evolution process. The adversary stops the chosen-message attack phase at a point of its choice and then gets to break-in. Throughout the adversary is allowed oracle access to the hash function H since the latter is modeled as a random oracle.

Thus the adversary F is viewed as functioning in three stages: the chosen message attack phase (cma); the break-in phase (breakin); and the forgery phase (forge). Its success probability in breaking $\mathsf{FSIG}[k,l,T]$ is evaluated by the following experiment:

It is understood above that in running F we first pick and fix coins for it, and also that we preserve its state across its various invocations. The chosen-message attack reflects the way the signature scheme is used. The adversary first gets access to an oracle for generating signatures under SK_1 . It queries this as often as it wants, and indicates it is done by outputting some value d. As long as d is not the special value breakin, we move into the next stage, providing the adversary an oracle for signing under the next key. Note that the process is

strictly ordered; once an adversary gives up the oracle for signing under SK_j , by moving into the next phase or breaking-in, it cannot obtain access to that oracle again. At some point the adversary decides to use its break-in privilege, and is returned the key SK_j of the stage in which it did this. (If it does not break-in by the last period, we give it the key of that period by default.) It will now try to forge signatures under SK_b for some b < j and is declared successful if the signature is valid and the message is new.

Following the concrete security paradigm used in [4], we associate to the scheme an *insecurity function* whose value is the maximum probability of being able to break the scheme, the maximum being over all adversary strategies restricted to resource bounds specified as arguments to the insecurity function. To make this precise we begin by letting $\mathbf{Succ}^{\mathrm{fwsig}}(\mathsf{FSIG}[k,l,T],F)$ denote the probability that the above experiment returns 1. (This is the probability that the adversary F is successful in breaking FSIG in the forward security sense.) Then the insecurity function is defined as

$$\mathbf{InSec}^{\mathrm{fwsig}}(\mathsf{FSIG}[k,l,T];t,q_{\mathrm{sig}},q_{\mathrm{hash}}) \ = \ \max_{F} \left\{ \ \mathbf{Succ}^{\mathrm{fwsig}}(\mathsf{FSIG}[k,l,T],F) \right\}.$$

The maximum here is over all F for which the following are true; the execution time of the above experiment, plus the size of the code of F, is at most t; F makes a total of at most q_{sig} queries to the signing oracles across all the stages; the total number of queries made to H in the execution of the experiment is at most q_{hash} .

Note the execution time is not just that of F but rather that of the entire experiment F-Forge(FSIG[k, l, T], F), so includes in particular the time to compute answers to oracle queries. The time to pick the hash function H is also included, measured dynamically by counting the time to reply to each hash oracle query by picking the response randomly at the time the query is made. Similarly, q_{hash} is the number of H queries in the experiment, not just the number made explicitly by F, so that it includes the H queries made by the signing and verifying algorithms. In particular, q_{hash} is always at least one due to the hash query made in the verification of the forgery.

The smaller the value taken by the insecurity function, the more secure the scheme. Our goal will be to upper bound the values taken by this insecurity function.

FACTORING. We will prove the security of our scheme by upper bounding its insecurity as a function of the probability of being able to factor the underlying modulus in time related to the insecurity parameters. To capture this, let $Fct(\cdot)$ be any algorithm that on input a number N, product of two primes, attempts to return its prime factors, and consider the experiment:

```
Experiment Factor (k, Fct)
```

```
Pick random, distinct k/2 bit primes p,q, each congruent to 3 \mod 4 N \leftarrow pq p',q' \leftarrow Fct(N) If p'q' = N and p' \neq 1 and q' \neq 1 then return 1 else return 0
```

Let $\mathbf{Succ}^{\mathrm{fac}}(Fct,k)$ denote the probability that the above experiment returns 1. Let $\mathbf{InSec}^{\mathrm{fac}}(k,t)$ denote the maximum value of $\mathbf{Succ}^{\mathrm{fac}}(k,Fct)$ over all algorithms Fct for which the running time of the above experiment plus the size of the description of Fct is at most t. This represents the maximum probability of being able to factor a k-bit Blum-Williams integer in time t. We assume factoring is hard, meaning $\mathbf{InSec}^{\mathrm{fac}}(k,t)$ is very low as long as t is below the running time of the best known factoring algorithm, namely about $2^{1.9k^{1/3} \lg(k)^{2/3}}$.

SECURITY OF OUR SIGNATURE SCHEME. We are able to prove that as long as the problem of factoring Blum-Williams integers is computationally intractable, it is computationally infeasible to break the forward security of our signature scheme. The following theorem provides a precise, quantitative statement, upper bounding the forward insecurity of our scheme as a function of the insecurity of factoring.

Theorem 1. Let $\mathsf{FSIG}[k, l, T]$ represent our key evolving signature scheme with parameters modulus size k, hash function output length l, and number of time periods T. Then for any t, any q_{sig} , and any $q_{\mathrm{hash}} \geq 1$

$$\begin{split} &\mathbf{InSec}^{\text{fwsig}}(\mathsf{FSIG}[k,l,T];t,q_{\text{sig}},q_{\text{hash}}) \\ &\leq q_{\text{hash}} \cdot T \cdot \left[2^{-l} + \sqrt{2lT \cdot \mathbf{InSec}^{\text{fac}}(k,t')} \right] + \frac{q_{\text{sig}} \cdot q_{\text{hash}}}{2^k} \;, \end{split}$$

where $t' = 2t + O(k^3 + k^2 l \lg(T))$.

The security parameter k must be chosen large enough that the $\mathbf{InSec}^{\mathrm{fac}}(k,t')$ (the probability of being able to factor the modulus in time t') is very low. The theorem then tells us that the probability of being able to compromise the forward security of the signature scheme is also low.

The theorem above proves the security of the scheme in a qualitative sense: certainly it implies that polynomial time adversaries have negligible advantage, which is the usual complexity based goal. However it also provides the concrete, or exact security, via the concrete indicated bound. In this case however, we know no attacks achieving a success matching our bound, and suspect our bound is not tight. Perhaps the concrete security can be improved, particularly with regard to the $q_{\text{hash}} \cdot T$ multiplicative factor.

The proof of this theorem is in two steps. Section 4 makes explicit an identification scheme that underlies the above signature scheme, and Lemma 1 shows that this identification scheme is forward secure as long as factoring is hard. Section 5 relates the forward security of our signature scheme to that of the identification scheme via Lemma 2. Putting these two lemmas together easily yields Theorem 1. We now turn to the lemmas.

4 Our Forward-Secure Identification Scheme

FORWARD-SECURE IDENTIFICATION. We consider the standard framework for a public-key based identification protocol. The prover is in possession of secret key

 SK_0 , while both the prover and the verifier are in possession of the corresponding public key PK. The prover wants to identify herself interactively to the verifier. A standard three-pass protocol will be used, consisting of a "commitment" Y from the prover, followed by a challenge c from the verifier, and finally an "answer" Z from the prover. The standard security concern is that an adversary (not in possession of the secret key) be able to identify itself to the prover under the public key PK of the legitimate prover.

We extend the setting to allow evolution of the secret key, enabling us to consider forward security. Thus the time over which the public key is valid is divided into T periods, and in period j the prover identifies itself using a secret key SK_j . The public key remains fixed, but the protocol and verification procedure depend on j, which takes the value of the current period, a value that all parties are agreed upon. Forward security means that an adversary in possession of SK_j should still find it computationally infeasible to identify itself to the verifier in a time period i previous to j.

This security measure having been stated should, however, at once create some puzzlement. Identification is an "on-line" activity. More specifically, verification is an on-line, one-time action. Once period i is over, there is no practical meaning to the idea of identifying oneself in period i; one cannot go back in time. So forward security is not a real security concern or attribute in identification. So why consider it? For us, forward-secure identification is only a convenient mathematical game or abstraction based on which we can analyze the forward security of our signature scheme. (And forward security of the signature scheme is certainly a real concern: the difference with identification is that for signatures verification can take place by anyone at any time after the signature is created.)

In other words, we can certainly define and consider a mathematically (if not practically) meaningful notion of forward security of an identification scheme, and analyze a scheme with regard to meeting this property. That is what we do here, for the purpose of proving Theorem 1.

THE SCHEME. Keys (both public and secret) are identical to those of our signature scheme, and are generated by the same procedure KG that we described in Section 3, so that at the start of the game the prover is in possession of the base secret key SK_0 . The public key PK is assumed to be in possession of the verifier. At the start of period j ($1 \le j \le T$) the prover begins by updating the previous secret key SK_{j-1} to the new secret key SK_j that she will use in period j. This process is done by the same update algorithm as for the signature scheme, specified in Section 3. The update having been performed, the key SK_{j-1} is deleted. During period j, the prover and verifier may engage in any number of identification protocols with each other, as need dictates. In these protocols, it is assumed both parties are aware of the value j indicating the current period. The prover uses key SK_j to identify herself. We depict in Figure 2 the identification protocol in period j, showing the steps taken by both the prover and the verifier.

The identification scheme is fully specified by key generation algorithm, key update algorithm, and the proving and verifying algorithms that underly

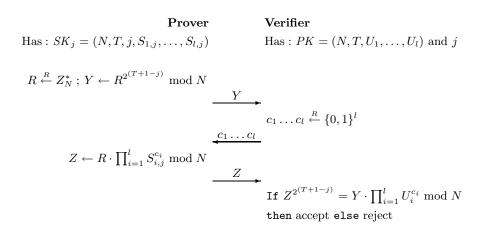


Fig. 2. Forward secure ID scheme: The protocol executed in period j. The prover has the secret key SK_j for period j and the verifier has the base public key PK and the value of j.

Figure 2. We let $\mathsf{FID}[k,l,T]$ denote our identification scheme when the parameters are fixed as indicated.

It should of course be the case that when the prover is honest, the verifier accepts. The proof of this is just like the proof of Proposition 1, the analogous issue for the signature scheme.

As should be clear from the identification protocol of Figure 2, the signing procedure under key SK_j that we defined in Section 3 is closely related to the identification protocol in period j: the former is derived from the latter by the standard process of specifying the challenge c via the value of a hash function on the message M and the commitment Y. This connection enables us to break the security analysis of the signature scheme into two parts; a forward-security analysis of the identification scheme, and an analysis of the preservation of forward-security of the "challenge hash" paradigm. This section is devoted to the first issue. We begin with the model.

SECURITY MODEL. The adversary in an identification scheme is called an impersonator and is denoted I. It knows the public key PK of the legitimate prover, and the current period j. We view I as functioning in two stages: the break-in phase (breakin) and the impersonation phase (imp). Its success probability in breaking FID is determined by the following experiment:

```
\begin{aligned} \textbf{Experiment} & \text{ F-Impersonate}(\mathsf{FID}[k,l,T],I) \\ & (PK,SK_0) \overset{R}{\leftarrow} KG(k,l,T) \\ & b \leftarrow I(\mathsf{breakin},PK) \\ & \text{ For } j=1,\dots,b \text{ do } SK_j \leftarrow Upd(SK_{j-1},j) \, ; \, j \leftarrow j+1 \text{ EndFor } \\ & Y,j \leftarrow I(\mathsf{imp},SK_b) \, ; \, c_1\dots c_l \overset{R}{\leftarrow} \{0,1\}^l \, ; \, Z \leftarrow I(c_1\dots c_l) \\ & \text{ If } Z^{2^{(T+1-j)}} = Y \cdot \prod_{i=1}^l U_i^{c_i} \text{ mod } N \text{ and } j < b \\ & \text{ then return } 1 \text{ else return } 0 \end{aligned}
```

In the break-in phase the adversary returns a time period $b \in \{1, ..., T\}$ as a break-in time, as a function of the public key. It will be returned the corresponding secret key SK_b , and now will try to successfully identify itself relative to some period j < b of its choice. Here it will try to impersonate the prover, so first chooses some commitment value Y. Upon being given the random challenge $c_1 ... c_l$ from the verifier, it returns an answer Z. It is considered successful if the verifier accepts the answer. Above, U_i is the i-th component of the public key PK. We let $\mathbf{Succ}^{\mathrm{fwid}}(\mathsf{FID}[k,l,T],I)$ denote the probability that the above experiment returns 1 and then let

$$\mathbf{InSec}^{\mathrm{fwid}}(\mathsf{FID}[k,l,T];t) \ = \ \max_{I} \, \{ \, \mathbf{Succ}^{\mathrm{fwid}}(\mathsf{FID}[k,l,T],I) \, \} \; ,$$

the maximum being over all I that for which the execution time of the above experiment, plus the size of the code of F, is at most t. As usual, the smaller the value taken by the insecurity function, the more secure the scheme. Our goal will be to upper bound the values taken by this insecurity function.

Notice that our model does not allow the adversary to first play the role of a cheating verifier, as does the standard model of identification. The reason is that we are only interested in applying this to the signature scheme, and there the challenge, being the output of the hash function, will be random, corresponding to an honest verifier. So it suffices to consider an honest verifier here.

SECURITY OF OUR IDENTIFICATION SCHEME. Next, we prove the security of our identification scheme by showing that breaking the forward security of our identification scheme is hard as long as the problem of factoring a Blum-Williams integer into its two prime factors is hard. The following lemma states a result which implies this:

Lemma 1. Let FID[k, l, T] represent our key evolving identification scheme with modulus size k, challenge length l, and number of time periods T. Then for any t

$$\mathbf{InSec}^{\mathrm{fwid}}(\mathsf{FID}[k,l,T];t) \leq 2^{-l} + \sqrt{2lT \cdot \mathbf{InSec}^{\mathrm{fac}}(k,t')} \;,$$

where
$$t' = 2t + O(k^3 + k^2 l \lg(T))$$
.

The first term in the bound represents the probability that the impersonator guesses the verifier's challenge, in which case it can of course succeed. The second term rules out any other attacks that are very different from simply trying to factor the modulus. The proof of Lemma 1 is omitted due to lack of space, and can be found in the full version of this paper [2] which is available on the web.

5 From Identification to Signatures

As we have noted, our signature scheme is derived from our identification scheme in the standard way, namely by having the signer specify the challenge c as a hash of the message M and commitment Y. In the standard setting we know that

this paradigm works, in the sense that if the identification scheme is secure then so is the signature scheme [17,15]. However we are not in the standard setting; we are considering an additional security property, namely forward security. The previous results do not address this. It turns out however that the hashing paradigm continues to work in the presence of forward security, in the following sense: if the identification scheme is forward secure, so is the derived signature scheme. This is a consequence of the following lemma:

Lemma 2. Let $\mathsf{FID}[k, l, T]$ and $\mathsf{FSIG}[k, l, T]$ represent, respectively our key evolving identification scheme and our key evolving signature scheme, with parameters k, l, T. Then for any t, any q_{sig} , and any $q_{\mathrm{hash}} \geq 1$

$$\mathbf{InSec}^{\mathrm{fwsig}}(\mathsf{FSIG}[k,l,T];t,q_{\mathrm{sig}},q_{\mathrm{hash}}) \leq q_{\mathrm{hash}} \cdot T \cdot \mathbf{InSec}^{\mathrm{fwid}}(\mathsf{FID},t') + \frac{q_{\mathrm{sig}} \cdot q_{\mathrm{hash}}}{2^k},$$

where
$$t' = t + O(k^2 l \lg(T))$$
.

The lemma says that if the forward insecurity of the identification scheme is small, so is that of the signature scheme. Combining Lemmas 2 and 1 proves Theorem 1, the main theorem saying our signature scheme is forward secure. The proof of Lemma 2 is omitted due to lack of space, and can be found in the full version of this paper [2] which is available on the web.

Acknowledgments

We thank the Crypto 98 program committee and Michel Abdalla for their comments, and Victor Shoup for helpful discussions. Mihir Bellare was supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering. Sara Miner was supported in part by a National Physical Sciences Consortium Fellowship and the above-mentioned grants of Bellare.

References

- R. Anderson, Invited lecture, Fourth Annual Conference on Computer and Communications Security, ACM, 1997.
- 2. M. Bellare and S. Miner, "A forward-secure digital signature scheme," Full version of this paper, available via http://www-cse.ucsd.edu/users/mihir.
- M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," Proceedings of the First Annual Conference on Computer and Communications Security, ACM, 1993.
- M. Bellare and P. Rogaway, "The exact security of digital signatures: How to sign with RSA and Rabin," Advances in Cryptology – Eurocrypt 96 Proceedings, Lec. Notes in Comp. Sci. Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.
- G. R. Blakley, "Safeguarding cryptographic keys." Proceedings of AFIPS 1979 National Computer Conference, AFIPS, 1979.
- L. Blum, M. Blum and M. Shub, "A simple unpredictable pseudo-random number generator," SIAM Journal on Computing Vol. 15, No. 2, 364-383, May 1986.

- Y. Desmedt and Y. Frankel, "Threshold cryptosystems." Advances in Cryptology Crypto 89 Proceedings, Lec. Notes in Comp. Sci. Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
- 8. W. DIFFIE, P. VAN OORSCHOT AND M. WIENER, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, 2, 107–125 (1992).
- U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," J. of Cryptology, 1(1988), 77-94.
- 10. A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," *Advances in Cryptology Crypto* 86 *Proceedings*, Lec. Notes in Comp. Sci. Vol. 263, A. Odlyzko ed., Springer-Verlag, 1986.
- S. GOLDWASSER, S. MICALI AND R. RIVEST, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal of Computing, Vol. 17, No. 2, pp. 281–308, April 1988.
- C. GÜNTHER, "An identity-based key-exchange protocol," Advances in Cryptology

 Eurocrypt 89 Proceedings, Lec. Notes in Comp. Sci. Vol. 434, J-J. Quisquater,
 J. Vandewille ed., Springer-Verlag, 1989.
- 13. S. Haber and W. Stornetta, "How to Time-Stamp a Digital Document," Advances in Cryptology Crypto 90 Proceedings, Lec. Notes in Comp. Sci. Vol. 537, A. J. Menezes and S. Vanstone ed., Springer-Verlag, 1990.
- A HERZBERG, M. JAKOBSSON, S. JARECKI, H KRAWCZYK AND M. YUNG, "Proactive public key and signature schemes," Proceedings of the Fourth Annual Conference on Computer and Communications Security, ACM, 1997.
- K. Ohta and T. Okamoto. "On concrete security treatment of signatures derived from identification," Advances in Cryptology Crypto 98 Proceedings, Lec. Notes in Comp. Sci. Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
- H. Ong and C. Schnorr, "Fast signature generation with a Fiat-Shamir like scheme," Advances in Cryptology – Eurocrypt 90 Proceedings, Lec. Notes in Comp. Sci. Vol. 473, I. Damgård ed., Springer-Verlag, 1990.
- 17. D. Pointcheval and J. Stern, "Security proofs for signature schemes," Advances in Cryptology Eurocrypt 96 Proceedings, Lec. Notes in Comp. Sci. Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.
- A. Shamir, "How to share a secret," Communications of the ACM, 22(1979), 612-613.
- V. Shoup, "On the security of a practical identification scheme," Advances in Cryptology – Eurocrypt 96 Proceedings, Lec. Notes in Comp. Sci. Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.
- 20. H. WILLIAMS, "A Modification of the RSA Public-key Encryption Procedure," *IEEE Transactions on Information Theory*, Vol. IT-26, No. 6, 1980, pp. 726–729.

Abuse-Free Optimistic Contract Signing

Juan A. Garay, Markus Jakobsson, and Philip MacKenzie

Information Sciences Research Center
Bell Laboratories
600 Mountain Ave
Murray Hill, NJ 07974
{garay,markusj,philmac}@research.bell-labs.com

Abstract. We introduce the notion of abuse-free distributed contract signing, that is, distributed contract signing in which no party ever can prove to a third party that he is capable of choosing whether to validate or invalidate the contract. Assume Alice and Bob are signing a contract. If the contract protocol they use is not abuse-free, then it is possible for one party, say Alice, at some point to convince a third party, Val, that Bob is committed to the contract, whereas she is not yet. Contract protocols with this property are therefore not favorable to Bob, as there is a risk that Alice does not really want to sign the contract with him, but only use his willingness to sign to get leverage for another contract. Most existing optimistic contract signing schemes are not abuse-free. (The only optimistic contract signing scheme to date that does not have this property is inefficient, and is only abuse-free against an off-line attacker.) We give an efficient abuse-free optimistic contract-signing protocol based on ideas introduced for designated verifier proofs (i.e., proofs for which only a designated verifier can be convinced). Our basic solution is for two parties. We show that straightforward extensions to n>2 party contracts do not work, and then show how to construct a three-party abuse-free optimistic contract-signing protocol.

An important technique we introduce is a type of signature we call a *private contract signature*. Roughly, these are designated verifier signatures that can be converted into universally-verifiable signatures by either the signing party or a trusted third party appointed by the signing party, whose identity and power to convert can be verified (without interaction) by the party who is the designated verifier.

1 Introduction

Contract signing is an important part of any business transaction, in particular in settings where participants do not trust each other to some extent already. Thus, the World Wide Web is probably the best example of a setting where contracts are needed. Still, even though a great amount of research has gone into developing methods for contract signing over a network such as the WWW, there is room for improvement. In this paper, we discuss some shortcomings of the present state of the art in digital contract signing, and propose a solution that

we prove has all the desired properties that we typically associate with fairness. Finally, we show how to extend our scheme to three participants without losing these properties.

Recently, considerable efforts have been devoted to develop protocols that mimic the features of "paper contract signing," especially fairness. A contract signing protocol is fair if at the end of the protocol, either both parties have valid signatures for a contract, or neither does. In some sense, this corresponds to the "simultaneity" property of traditional paper contract signing. That is, a paper contract is generally signed by both parties at the same place and at the same time, and thus is fair.

Early work on electronic contract signing, or more generally, fair exchange of secrets/signatures, focused on the gradual release of secrets to obtain simultaneity, and thus fairness [9,25,30] (see [21] for more recent results). The idea is that if each party alternately releases a small portion of the secret, then neither party has a considerable advantage over the other. Unfortunately, such a solution has several drawbacks. Apart from being expensive in terms of computation and communication, it has the problem in real situations of uncertain termination: if the protocol stops prematurely and one of the participants does not receive a message, he will never be sure whether the other party is continuing with the protocol, or has stopped—and perhaps even has engaged in another contract signing protocol!

An alternative approach to achieving fairness instead of relying on the gradual release of secrets has been to use a trusted third party (TTP). A TTP is essentially a judge that can be called in to handle disputes between contract signers. This was the case in the early work of [24], where the contract is only valid if a "center of cancelation" does not object. The TTP can be on-line in the sense of mediating after every exchange as in [16,22,28], at the expense of the TTP becoming a potential bottleneck, or off-line, meaning that it only gets involved when something goes wrong (e.g., a participant attempts to cheat, or simply crashes, or the communication delays between the participants are intolerably high, etc.). The latter approach has been called optimistic [2], and fair contract signing protocols have been developed in this model [2,3]. Although the TTP is (by definition) trusted, it is, in some protocols, possible for the TTP to be accountable for his actions. That is, it can be determined if the TTP misbehaved.

Most of these protocols have the conceptual drawback of allowing one of the parties at some time to convince a third party he has a potentially signed contract. That is, this participant can demonstrate that the other party is committed (to some extent) to the agreement, without himself being committed. Ultimately, of course, we want neither of the participants to have this ability until the contract has been signed by both parties.

In this sense, it is possible to distinguish between two types of attacks, which we call *on-line* vs. *off-line*. In an on-line attack, the cheating contract signer and a third party may interact in an arbitrary fashion during the contract signing protocol, in order for the third party to be convinced of the other signer's

willingness to sign. In an off-line attack, however, this is not allowed, and the cheating contract signer may only send the third party one message. Clearly, the former is a stronger form of attack. All previously proposed contract signing protocols are vulnerable against the on-line attack, and all proposed optimistic contract signing protocols – but one – are vulnerable against the off-line attack. Our proposed scheme is secure against both types of attacks. A third – and even stronger – attack allows the cheating signer to cooperate with the third party from the "beginning of time". The attack involves the sharing of the cheating signer's secret key by the cheating signer and the third party (i.e., the key is stored in a way such that neither of them can reconstruct it on their own.) Such an attack must be initiated before the certification of the corresponding public key. As in [31], we have to assume that the certification process is done in an "isolated" manner, and that the certified party be required to prove knowledge of his secret key before he becomes certified, thereby preventing successful "identity sharing" of this type. We believe that this is the only defense against such an "ultimate" attack, apart from mere social constraints that make very careful planning and administering of attacks "from the beginning of time" less likely to take place.

Our results. Our contributions are twofold:

- In this paper we introduce the notion of "abuse freeness" to the area of electronic contract signing, and present an efficient abuse-free optimistic contract signing protocol. We also show how to extend the scheme to three parties without losing these properties.
- To build the protocols we use a new type of signature that we call private contract signatures (PCS). Roughly, these are designated verifier signatures that can be converted into universally-verifiable signatures by either the signing party or a trusted third party appointed by the signing party, whose identity and power to convert can be verified (without interaction) by the party who is the designated verifier. We give a complete formalization of these signatures, and present an efficient discrete log-based scheme which uses new boolean combinations of proofs of membership and knowledge. We prove the scheme to be as secure as the Decisional Diffie-Hellman problem.

Related work. Contract signing is part of the broader problem of fair exchange [10,16,28,33,2]. More specifically, it can be considered fair exchange of digital signatures [3]. The different existing approaches to contract signing were already described above. The term "contract signing" was first introduced in [8]. The first optimistic scheme in the sense defined above was based on the gradual increase of privilege [7]: as the computation evolves, the probability of a contract being valid gradually increases from 0 to 1. This solution has several shortcomings, as it requires termination detection by the third party (i.e., synchronous system), and one of the parties might be "privileged," in the sense of only one of them being able to finalize the contract; this happens with a non-negligible probability.

The most closely related work to ours is Asokan, Shoup, and Waidner [3]. They present an optimistic protocol for fair exchange of digital signatures. They

allow any of a variety of signature schemes, and prove security of their protocol. It is not mentioned, but one of their protocols is off-line abuse-free. However, their protocol is inefficient, due to the use of expensive cut-and-choose techniques. More specifically, to obtain a failure probability of 2^{-k} , they require 2k+4 exponentiations for both prover and verifier. In contrast, we require 32 exponentiations for the prover and the verifier, regardless of the security parameter. Also, their scheme does not allow the trustee to be held accountable for his actions, meaning that if the trustee were to misbehave, it would not be possible for the participants to show to a judge that this must have happened. They present another fair optimistic contract signing protocol that is more efficient, but does not have the abuse-free property, nor the ability to allow any type of signature scheme.

The abuse-free contract signing protocol we present is more efficient (in number of rounds, communication bits, and computation), than the only published off-line abuse-free protocol so far, namely the one in [3], and provides for trustee accountability. Our scheme is both off-line and on-line abuse-free. However, our protocol restricts the participants to a certain type of signature.

The private contract signatures that we introduce exhibit some similarities with designated-confirmer signatures and convertible undeniable signatures, two notions first suggested by Chaum [12]. A designated-confirmer signature is an undeniable signature [13,15] that can be proven valid by a so-called confirmer, an entity appointed by the signer and given a secret key for verification by the same. A convertible undeniable signature is an undeniable signature that can be converted to a standard (i.e., self-authenticating) signature, This can be done either by the signer or, in the setting of designated confirmer signatures, by a trusted third party. Although Chaum's scheme was broken by Michels et al. [34], its conceptual ideas are still intact, and alternative solutions were suggested in [19,34].

One major difference between the concept of convertible undeniable signatures and private contract signatures is that the latter allow an optimistic approach in that the receiver knows beforehand that the trusted third party can convert the signature. In contrast, a convertible undeniable signature needs a verification session, such as an interactive undeniable signature verification [13,15]. Technically, they differ in that the signer of a convertible undeniable signature needs to give a secret key to the trusted third party, whereas a private contract signature is performed relative to the public key of the trusted third party, who does not need to obtain any secret key from the signer. One could say that a private contract signature is a special case of a "designated-confirmer convertible signature," namely, one in which the receiver knows that the contents correspond to a valid publicly-verifiable signature, but can't convince anybody else about this without the help of either the signer or the trusted third party.

¹ Batch computation methods (e.g., [32,5]) can be applied to both the schemes, and although they can be applied to a larger extent to [3], their scheme remains noticeably more expensive than ours for reasonable security parameters.

Our work is also closely related to work on designated-verifier proofs, which were introduced by Jakobsson et al. [31]. A designated-verifier proof of a statement Θ is a proof with the property that it will convince nobody but the so-called "designated verifier" (who is selected by the prover) of the correctness of Θ . Furthermore, the designated verifier will accept a valid such proof and reject an invalid such proof with an overwhelming probability². We call a non-interactive designated-verifier proof a designated verifier signature.

A private contract signature is a designated-verifier signature with the property that it may be converted to a standard, self-authenticating signature, drawing upon the ideas of designated *converter* proofs and convertible undeniable signatures. This conversion can be performed by two participants only: the participant who generated the signature, and a trusted third party, who is appointed by the participant who generated the signature, and whose identity can be determined from the transcript constituting the private contract signature. However, as already mentioned, the initial receiver of the signature need not interact with anybody in order to determine whether the transcript he has received can always be converted into a standard signature.

Finally, we show that natural extensions of an abuse-free protocol to three-party contract signing do not seem to allow abuse-freeness. However, we present a three-party fair abuse-free optimistic contract-signing protocol obtained by careful design of a (much more intricate) protocol.³ Independently of our work, *multi-party* contract signing protocols were given in [1,4] While not abuse-free, the protocol in [4] is simpler than ours, resulting in a more elegant and efficient general solution.

2 Model and Definitions

Our basic model is similar to that of [3]. We have set of participants $S = \{P_1, P_2, \dots, P_n\}$, and a trusted third party T. Participants may be correct or faulty (Byzantine). Formally, the participants and the trusted third party are modeled by probabilistic interactive Turing machines. We assume all participants have public/private keys which will be specified later. T acts like a server, responding (atomically) to requests from the participants, with responses defined later. We assume that communication between any participants and T is over a private channel. We will first concentrate on the case of two participants, A and B.

The network model we consider is the same as in [3]. Namely, an asynchronous communication model with no global clocks, where messages can be delayed arbitrarily [27], but with messages sent between correct participants and the trusted third party guaranteed to be delivered eventually. In general, we assume

² Depending on the proof structure, this probability might be a constant fraction, in which case the standard method of repeating the proof can be applied.

³ In fact, this has been simplified and extended to a general n-party contract signing [29].

an adversary may schedule messages, and possibly insert its own messages into the network.

A and B wish to "sign" a contract m.⁴ By signing a contract, we mean providing public, non-repudiable evidence that they have agreed to the contract. The type of evidence is either universally agreed upon, or is part of the contract m itself. For instance, valid evidence may include either valid signatures by A and B on m, or valid signatures by A and B on the concatenation of m and the public key of T, signed together by T.

Obviously, in the asynchronous model of [27], an adversary may prevent a contract from being signed simply by delaying all messages between players. Thus in order to force contract signing protocols to be non-trivial, we specify a completeness condition using a slightly restricted adversary. An optimistic contract signing protocol is complete if the (slightly restricted) adversary cannot prevent a set of correct participants from obtaining a valid signature on a contract. This adversary has signing oracles that can be queried on any message except m, can interact with T, and can arbitrarily schedule messages from the participants to T. However, it cannot delay messages between the correct participants enough to cause any timeouts.

An optimistic contract signing protocol is fair if

- (1) it is impossible for a corrupted participant (say A^*) to obtain a valid contract without allowing the remaining participant (say B), to also obtain a valid contract;
- (2) once a correct participant obtains a cancelation message from the TTP T, it is impossible for any other participant to obtain a valid contract; and
- (3) every correct participant is guaranteed to complete the protocol. Effectively, condition (2) provides a *persistency* condition, stating that correct participants cannot have their outcomes overturned by other participants.

Finally, an optimistic contract signing protocol is *abuse-free* if it is impossible for a single player (say A^*) at any point in the protocol to be able to prove to an outside party that he has the power to terminate (abort) or successfully complete the contract.

We say an optimistic contract signing protocol is *secure* if it is fair, complete, and abuse-free.

3 Cryptographic Tools and Techniques

For a prime q, we will work in a group G_q of order q with generator g. For a specific example, we could use a group form by integer modular arithmetic as follows. Let p be a prime of the form lq + 1 for some value l co-prime to q, and let Z_p^* be the group, with g a generator of order q in Z_p^* . Then G_q would be the subgroup generated by g.

⁴ In general, A and B might need to sign two different pieces of text, m and m'. Extension to this case is trivial.

We assume the hardness of the *Diffie-Hellman decision problem* (DDH). In this problem, it is the goal of the p-time adversary to distinguish the following two distributions with a non-negligible advantage over a random guess:

1. (g_1, g_2, y_1, y_2) with $g_1, g_2, y_1, y_2 \in_R G_q$, and 2. (g_1, g_2, g_1^r, g_2^r) with $g_1, g_2 \in_R G_q$ and $r \in_R Z_q$.

ElGamal encryption [23] is a public-key encryption scheme, in which it is assumed that the recipient Bob is associated with a public key y and Bob has a private key x such that $g^x = y$. To generate an ElGamal encryption of a value $m \in G_q$ for Bob, a sender Alice generates $r \in_R Z_q$, and sends $(y^r m, g^r)$. Bob decrypts an incoming message (a, b) by computing a/b^x .

The security of ElGamal encryption has been shown to equal the security of the Diffie-Hellman decision problem.

3.1 Non-interactive Proofs of Knowledge

Our protocols will require certain non-interactive proofs. In this paper, these proofs will basically be the non-interactive versions of Σ -protocols [20,26] (i.e., Schnorr-like protocols [36]). However, we use a general notion of Σ -protocols [17] which encompass both proofs of knowledge and membership. We refer the reader to [17] for the full technical definitions.

We also use proofs of conjunctions and disjunctions (i.e., ANDs and ORs) of certain statements which have known non-interactive proofs [18]. To simplify notation, we will often state these as conjunctions and disjunctions of the specific proofs (instead of the statements), which will indicate proofs of conjunctions and disjunctions of the statements built on those specific proofs. (More formally, a boolean function b() over some non-interactive proofs will denote the non-interactive proof corresponding to the Σ -protocol that is obtained from the boolean function b() applied to the Σ -protocols corresponding to the individual proofs.)

Finally, we use designated verifier proofs/signatures ([31], see also [12]), which are built using conjunctions and disjunctions.

Proof of Ciphertext Contents. Let (a, b) be an ElGamal ciphertext. We want a protocol for proving ciphertext contents, i.e., a protocol for proving that a ciphertext (a, b) corresponds to a plaintext m. We want to consider such proofs that can be disjunctively composed (which excludes some of the simplest approaches), and for this we will use a Σ -protocol.

Specifically, we want to prove that $(a, b) = (y^r m, g^r)$ for some value r and a specified public key y and plaintext m. This can be done by the party who performed the encryption (and who therefore knows the value r) and also by the party who knows the secret key corresponding to y (since this party can decrypt the ciphertext.) In the first case, this is the same as proving that $\log_y(a/m) = \log_g b$, in the second case it is proving that $\log_b(a/m) = \log_g y$. We will call the corresponding non-interactive versions of the Σ -protocols " $m = \mathrm{CC}_E(a, b, y)$ "

(meaning "ciphertext contents – by encryptor") and " $m = CC_D(a, b, y)$ " (meaning "ciphertext contents – by decryptor".)

The details of the proof of ciphertext contents by encryptor or decryptor are straightforward and omitted.

DL-based signatures: We can use a variety of signature schemes which consist to proving knowledge of a discrete log (a private key corresponding to a public key) made relative to a message m One such scheme is the Schnorr signature scheme [36]. We will denote such a scheme DL-SIG, and will denote a non-interactive proof by a party with public key y relative to a message m by DL-SIG $_y(m)$.

We omit the details of the interactive and non-interactive proofs for the Schnorr scheme.

Designated Verifier Proofs/Signatures. A designated verifier proof is a proof that is convincing to the designated verifier, but not to any other party. A proof (such as a witness hiding proof, zero-knowledge proof, or Schnorr proof) of an assertion Θ can be transformed into a a proof for a designated verifier Bob by modifying what is being proved to " Θ or 'I know Bob's private key'". If Bob shows this proof to any other party, they would not be convinced of the veracity of Θ , since Bob could have constructed the proof himself even if Θ were not true, simply using knowledge of his private key.

An example of a designated verifier signature was given in the previous section: "DL-SIG $_y(m)$ VDL-SIG $_{y'}(m)$." Say Alice knows the private key x associated with y and Bob knows the private key x' associated with y'. If Alice gives this proof to Bob, Bob is convinced of Alice's knowledge of x and signature on m, but if he shows this to any other party, they would not necessarily be convinced that Alice signed m.

4 Private Contract Signatures

An important technique we introduce is a type of signature we call a *private contract signature*. Roughly, these are designated verifier signatures that can be converted into universally-verifiable signatures by either the signing party or a trusted third party appointed by the signing party, whose identity can be verified by the party who is the designated verifier. More formally:

Definition 1. A private contract signature (PCS) scheme Σ is a tuple of probabilistic polynomial-time algorithms {PCS-Sign, S-Convert, TP-Convert, PCS-Ver, S-Ver, TP-Ver} defined as follows, and having the security properties defined below.

(1) PCS-Sign executed by party A on m for B with respect to third party T, denoted PCS-Sign $_A(m,B,T)$, outputs a private contract signature, denoted PCS $_A(m,B,T)$. A private contract signature can be verified using PCS-Ver, i.e.,

$$\mathsf{PCS\text{-}Ver}(m,A,B,T,S) = \begin{cases} \text{true } \textit{if } S = \mathsf{PCS}_A(m,B,T); \\ \text{false } \textit{otherwise}. \end{cases}$$

- (2) S-Convert executed by A on a private contract signature $S = PCS_A(m, B, T)$ generated by A, denoted S-Convert_A(S), produces a universally-verifiable signature by A on m, S-Sig_A(m).
- (3) TP-Convert executed by T on a private contract signature $S = PCS_A(m, B, T)$, denoted TP-Convert $_T(S)$, produces a universally-verifiable signature by A on m, TP-Sig $_A(m)$.
- (4) $\operatorname{S-Sig}_A(m)$ can be verified using S-Ver, and $\operatorname{TP-Sig}_A(m)$ can be verified using TP-Ver, i.e.,

$$\mathsf{S\text{-}Ver}(m,A,T,S) = \begin{cases} \text{true } \textit{if } S = \mathsf{S\text{-}Sig}_A(m); \\ \text{false } \textit{otherwise}; \end{cases}$$

and

$$\mathsf{TP\text{-}Ver}(m,A,T,S) = \begin{cases} \mathsf{true} \ if \ S = \mathsf{TP\text{-}Sig}_A(m); \\ \mathsf{false} \ otherwise. \end{cases}$$

The security properties of a PCS scheme are:

- (1) **Unforgeability of** $PCS_A(m, B, T)$: For any m, it is infeasible for anyone but A or B to produce T and S such that PCS-Ver(m, A, B, T, S) = true.
- (2) **Designated verifier property of** $PCS_A(m, B, T)$: For any B, there is a polynomial-time algorithm FakeSign such that for any m, A, and T, $FakeSign_B(m, A, T)$ outputs S, where PCS-Ver(m, A, B, T, S) = true.
- (3) Unforgeability of $S-Sig_A(m)$ and $TP-Sig_A(m)$: For any m, A, B, T, assuming
 - $P = PCS_A(m, B, T)$ is known and was produced by PCS-Sign_A(m, B, T), it is infeasible
 - (3.1) for anyone but A or T to produce S such that S-Ver(m, A, T, S) = true and TP-Ver(m, A, T, S) = true;
 - (3.2) for anyone but A to produce S such that S-Ver(m, A, T, S) = trueand TP-Ver(m, A, T, S) = false; and
 - (3.3) for anyone but T to produce S such that $\mathsf{TP\text{-}Ver}(m,A,T,S) = \mathsf{true}$ and $\mathsf{S\text{-}Ver}(m,A,T,S) = \mathsf{false}$.

Definition 2. A PCS scheme Σ is third party-accountable if for any PCS_A(m, B, T), the distributions of S-Sig_A(m) and TP-Sig_A(m), produced by S-Convert and TP-Convert, respectively, are disjoint.

That is, it is impossible to have $\mathsf{S-Ver}(m,A,T,S) = \mathsf{TP-Ver}(m,A,T,S) = \mathsf{true}$ for any S, and thus it is possible for a verifier to distinguish whether the conversion was performed by the signatory or by the third party. This property might be useful if the signatories are concerned about the third party's trustworthiness. A somewhat complementary property is the following.

Definition 3. A PCS scheme Σ is third party-invisible if for any PCS_A(m, B,T), the distributions of S-Sig_A(m) and TP-Sig_A(m), produced by S-Convert and TP-Convert, respectively, are identical.

In other words, for any S, S-Ver(m,A,T,S) = TP-Ver(m,A,T,S), meaning no one can determine if a conversion was performed by the original signer or the trustee. This property might be useful in some scenarios where signatories may not want possible "bad publicity" associated with a contract needing to be converted by the third party.

Note that the scheme given in [3] that uses verifiable encryption is actually a PCS scheme, with the TP-invisibility property. That scheme is inefficient, requiring cut-and-choose techniques, but it allows for a variety of types of signatures. We now present an efficient PCS scheme in which the signatures are of a specific form.

4.1 An Efficient Discrete Log-based PCS Scheme

Say A has public key y_A and private key x_A , where $y_A = g^{x_A}$. Similarly, B has public key/private key pair (y_B, x_B) , and T has public key/private key pair (y_T, x_T) . The intuition is as follows. In order for A to generate a PCS on m for B, she sends B a proof of the statement

"X is a T-encryption of "1" AND I can sign
$$m$$
 as A OR X is a T-encryption of "2" AND I can sign m as B "

where X is some value, and T-encryption denotes a message encrypted with T's public key. A can do this because she can perform a T-encryption of "1" to generate X, and she can sign m herself. Upon receiving this proof, B can verify its correctness, but he is not able to transfer it to anybody else, since he could have generated a similar proof himself.

A TP-accountable conversion of the above PCS by A can be done by A giving a proof that X is a T-encryption of "1" (cf. Section 3.1). The TP-accountable conversion of the PCS by T can be done similarly, i.e., by T giving a proof that X is a T-encryption of "1." The accountability will arise from the fact that the proofs given by A and T are of a different form. For a TP-invisible conversion of the PCS, both A and T output the disjunction of the two proofs above. We now define the scheme formally by describing the algorithms.

- PCS-Sign_A(m, B, T) works as follows. A generates an ElGamal encryption (a, b) of 1 for T and outputs $PCS_A(m, B, T) = \langle (a, b), P \rangle$ where P is the following proof:

$$([1 = \mathrm{CC}_E(a, b, y_T)] \wedge \mathrm{DL\text{-}SIG}_{y_A}(m)) \vee ([2 = \mathrm{CC}_E(a, b, y_T)] \wedge \mathrm{DL\text{-}SIG}_{y_B}(m)))$$

- S-Convert_A(S), where $S = PCS_A(m, B, T)$, works as follows. For a trustee-invisible signature, A outputs S-Sig_A(m) = $\langle S, P \rangle$ where P is the following proof:

$$[1 = CC_E(a, b, y_T)] \lor [1 = CC_D(a, b, y_T)].$$

For a non-trustee-invisible signature, A outputs $\mathrm{S\text{-}Sig}_A(m)=\langle S,P\rangle$ where P is the following proof:

$$1 = CC_E(a, b, y_T).$$

- TP-Convert_T(S), where $S = PCS_A(m, B, T)$, works as follows. For a trustee-invisible signature, T outputs TP-Sig_A(m) = $\langle S, P \rangle$ where P is the following proof:

$$[1 = CC_E(a, b, y_T)] \lor [1 = CC_D(a, b, y_T)].$$

For a non-trustee-invisible signature, T outputs $\operatorname{TP-Sig}_A(m) = \langle S, P \rangle$ where P is the following proof:

$$1 = CC_D(a, b, y_T).$$

Theorem 1. Assuming the security of DDH, the above signature scheme is a PCS scheme (in the random oracle model).

Proof Sketch: First we show unforgeability of $PCS_A(m, B, T) = \langle (a, b), P \rangle$. Say a polynomial-time party F could forge with non-negligible probability. Then we could use F to find x_A or x_B in polynomial-time with non-negligible probability, as follows. Assuming that the challenge length is superpolynomial in k, the Forking Lemma from Pointcheval and Stern [35] shows that in polynomial time we can find two transcripts (a, c, z) and (a, c', z') for the Σ -protocol S corresponding to P. By the definition of S, we can extract x_A or x_B , which contradicts DLA, and hence contradicts DDH.

The designated verifier property follows from the fact that by DDH, it is infeasible to distinguish an ElGamal encryption of 2 from an ElGamal encryption of 1, and the fact that B has a polynomial time algorithm to generate $PCS_A(m, B, T)$. For the latter, first B generates an ElGamal encryption (a, b) of 2 using public key y_T . Then it can construct the proof P since it knows the secret value used in the encryption and it knows x_B , and thus can play the part of the prover in the Σ -protocol corresponding to the proof P needed for $PCS_A(m, B, T)$.

Now we show unforgeability of $\operatorname{S-Sig}_A(m) = \langle S, P \rangle$, where $S = \operatorname{PCS}_A(m,B,T) = \langle (a,b),P_S \rangle$. (The proof of unforgeability of $\operatorname{TP-Sig}_A(m)$ is similar.) First, say a polynomial-time party F' could construct P with nonnegligible probability such that (a,b) is not of the form (y_t^α,g^α) (i.e., (a,b) is not the encryption of "1"). Then by the Forking lemma, in polynomial time we can find two transcripts (a,c,z) and (a,c',z') for the Σ -protocol S' corresponding to P. But this is impossible, since the input is not in the language.

Thus, no polynomial-time party F' could construct P unless (a,b) is of the form $(y_t^{\alpha}, g^{\alpha})$. Now say a polynomial-time party F could forge $S\text{-}\mathrm{Sig}_A(m) = \langle S, P \rangle$ with non-negligible probability. There are two cases.

1. F has a non-negligible probability of forging S-Sig_A(m) = $\langle S, P \rangle$ for an S generated by A. In this case, we can break DDH. Given DDH instance (g, h, r_1, r_2) , set $y_T = h$, $b = r_1$ and $a = r_2$, and produce $PCS_A(m, B, T)$ (simulating the random oracle h also, in order to be able to construct the PCS proof). If the DDH instance is a true Diffie-Hellman quadruple, F constructs a valid P with non-negligible probability. Otherwise F cannot.

If F constructs a valid P, we guess that the DDH instance is a true Diffie-Hellman quadruple, and otherwise, with probability 1/2 we guess that it is a true Diffie-Hellman quadruple. Thus we guess with a non-negligible advantage.

2. F has a non-negligible probability of forging S-Sig_A $(m) = \langle S, P \rangle$ for an S not generated. Then F has a non-negligible probability of forging a PCS, which is a contradiction to our argument above.

Remark. We note that it may be possible to obtain a similar result using techniques involving encryption of signatures, and proofs that the resulting ciphertexts contain valid signatures. Efficient protocols for proving that the contents of a ciphertext is a valid signature were presented in [19] for ElGamal signatures, and [11] for RSA signatures. Combining this with the methods in [31] may result in alternative implementations.

5 Abuse-Free Contract Signing

In this section, we show how to use the PCS scheme to design an abuse-free contract signing protocol. First we give a high level description. The main protocol has an alternating structure, with a previously agreed-upon initiator, say, party A. If no problems occur, A sends a private contract signature (PCS) of the message to be signed to B, and B responds with his PCS. Then A converts her PCS into a universally-verifiable signature, and then B does the same. If problems occur, A or B may run protocols to abort or resolve the contract signing, depending on what step has been attained in the protocol. The basic, failure-free protocol is shown in Figure 1.

This is very similar to the protocol of [3], except that the primitives underlying our protocol are different, and provide different functionality. We now describe the protocol(s) in more detail. We use $[\alpha]_X$, for $X \in \{A, B, T\}$, to denote the string α concatenated with a signature of α under X's public key.

Signatory
$$A$$
 Signatory B
$$\frac{\operatorname{PCS}_A(m,B,T)}{\operatorname{PCS}_B(m,A,T)} \text{ If } \neg \operatorname{ok, quit}$$

$$\frac{\operatorname{PCS}_B(m,A,T)}{\operatorname{S-Sig}_A(m)} \text{ If } \neg \operatorname{ok, resolve}$$

$$\frac{\operatorname{S-Sig}_B(m)}{\operatorname{S-Sig}_B(m)}$$

Fig. 1. The basic abuse-free contract signing protocol for two parties

Main Protocol

- 1. $A \text{ runs PCS-Sign}_A((m,1), B, T)$ and sends the result, $PCS_A((m,1), B, T)$, to B.
- 2. If B receives a valid $PCS_A((m,1), B, T)$, it runs $PCS-Sign_B((m,2), A, T)$ and sends the result $PCS_B((m,2), A, T)$, to A. Otherwise B simply quits.
- 3. If A receives a valid response $PCS_B((m, 2), A, T)$, it runs $S-Convert_A(PCS_A((m, 1), B, T))$ and sends the result, $S-Sig_A((m, 1))$, to B. Otherwise A runs the **Abort** protocol.
- 4. If B receives a valid response $\operatorname{S-Sig}_A((m,1))$, it runs $\operatorname{S-Convert}_B(\operatorname{PCS}_B((m,2),A,T))$ and sends the result, $\operatorname{S-Sig}_B((m,2))$, to A. Otherwise B runs B-Resolve.
- 5. If A does not receive a valid response $S-Sig_B((m,2))$, she runs resolve

Protocol Abort: To abort, A sends $[m, A, B, abort]_A$ to T.

- 1. If the signature is correct, and neither A nor B have resolved, T sends $[[m, A, B, abort]_A]_T$ back to A, and stores this.
- 2. If either A or B has resolved, it sends the corresponding stored value (i.e., $S\text{-}\mathrm{Sig}_A((m,1))$ or $S\text{-}\mathrm{Sig}_B((m,2))$, resp.—see below).

Protocol B-Resolve: For B to resolve, it runs S-Convert_B($PCS_B((m, 2), A, T)$) to produce S- $Sig_B((m, 2))$, and sends the message

$$(\mathrm{PCS}_A((m,1),B,T),\mathrm{S\text{-}Sig}_B((m,2)))$$

to T. T checks to make sure the second half corresponds to the first half (i.e., m is the same, etc.), and checks that the second half is valid. If so, then it decides what to do as follows.

- 1. If A has aborted, it sends the stored copy of $[[m, A, B, abort]_A]_T$ to B.
- 2. If A has resolved, it sends the stored copy of $\operatorname{S-Sig}_A((m,1))$ to B.
- 3. Otherwise, it sends $\mathsf{TP\text{-}Convert}_T(\mathsf{PCS}_A((m,1),B,T))$ to B and stores this.

Protocol A-Resolve: For A to resolve, it runs S-Convert_A($PCS_A((m, 1), B, T)$) to produce S-Sig_A((m, 1)), and sends the message

$$(\operatorname{S-Sig}_A((m,1)),\operatorname{PCS}_B((m,2),A,T))$$

to T. T checks to make sure the first half corresponds to the second half (i.e., m is the same, etc.), and checks that the first half is valid. If so, then it decides what to do as follows.

- 1. If A has already aborted, it sends the stored copy of $[[m, A, B, T]_A]_T$.
- 2. If B has resolved, it sends the stored copy of S-Sig_B((m, 2)).
- 3. If B has not resolved, it sends $\mathsf{TP\text{-}Convert}_T(\mathsf{PCS}_B((m,2),A,T))$ to A and stores this.

Recall that the S-Convert and TP-Convert procedures can result in either TP-invisibility or TP-accountability.

Disputes: Disputes concerning contracts are handled as follows:

- 1. If one party shows $[[m, A, B, abort]_A]_T$ and the other shows $(S-Sig_B^B(m), S-Sig_A^A(m))$, then the contract is valid, since A must have called **Abort** after sending $S-Sig_A^A(m)$, and thus after ratifying the contract.
- 2. If one party shows $[[m, A, B, abort]_A]_T$ and the other shows a contract with $\text{TP-Sig}_A(m)$ or $\text{TP-Sig}_B(m)$, then if the TP-accountable conversions were used, T must have cheated.

If we assume that TP-invisibility is used and the third party is trusted, only the first case above may occur.

Theorem 2. Assuming the security of DDH, the protocol above is a secure optimistic contract signing protocol, with TTP invisibility.

Proof. Recall that we say that an optimistic contract signing protocol is secure if it is complete, fair and abuse-free.

Complete: Completeness follows directly from the PCS definition.

Fairness for A: Say A is honest and does not obtain a valid contract. First consider the case in which A calls **Abort** after sending $PCS_A(m, B, T)$. Since she does not obtain a valid contract, she must receive $[[m, A, B, abort]_A]_T$. Then by the security of PCS, B is not able to construct S-Sig $_A^A(m)$ or S-Sig $_A^T(m)$. The only other case is when A sends S-Sig $_A^A(m)$ to B. But then A either receives S-Sig $_B^B(m)$ or calls A-Resolve to receive S-Sig $_B^T(m)$, and thus this case is impossible.

Fairness for B: Say B is honest and does not obtain a valid contract. This implies A never sent S- $Sig_A^A(m)$. Then at some point B must have called B-**Resolve** and obtained $[[m, A, B, abort]_A]_T$. Then A only received $PCS_B(m, A, T)$, and by the security of PCS, could not have a valid contract.

If T is not honest, then it is easy to see that this can be detected in the TP-accountable version of the protocol, but not necessarily in the TP-invisible version.

Abuse-freeness for A: We must show that B does not obtain publicly verifiable information about (honest) A signing the contract until B is also bound by the contract. This follows by the designated verifier property of $PCS_A(m, B, T)$, and the fact that once B receives $S-Sig_A^A(m)$, B must have sent $PCS_B(m, A, T)$, and B is not able to abort. Thus either A will receive $S-Sig_B^B(m)$ from B, or A will perform A-Resolve to obtain $S-Sig_B^T(m)$. Either way, A receives a valid contract.

Abuse-freeness for B: We must show that A does not obtain publicly verifiable information about (honest) B signing the contract until A is also bound by the contract. This follows by the designated verifier property of $PCS_B(m, A, T)$, and the fact that once A receives $S-Sig_B^B(m)$, A must have sent $S-Sig_A^B(m, B, T)$, and thus B already has a valid contract.

Remark: We have not distinguished between off-line and on-line attacks in the above proof sketch. In fact, we only need to consider on-line attacks, as the off-line attack is a special case of the on-line attack. When we refer to the designated verifier property above, we mean the property that unless the secret key of the receiver of a proof is not known by the same, it is under no circumstances possible for this receiver to convince a third party of the validity of the proof, whether they interact or not. Therefore, it follows that the validity of the proof/signature cannot be verified by said third party, but only by the party designated by the prover/signer.

TP Invisibility: Straightforward from the definition.

Similarly, we can show

Theorem 3. Assuming the security of DDH, the protocol variation above is a secure optimistic contract signing protocol with TTP accountability.

6 Three-Party Contracts

In this section we consider contract signing by n > 2 parties. Natural extensions to three-party contract signing do not seem to allow abuse-freeness. One problem faced in extensions to three parties is that the two party protocol is asymmetric: A has the power to abort, but B does not. With a straightforward extension to three parties, it is unclear whether to let the "middle" party abort or not, and how to maintain fairness and abuse-freeness in either case. (We leave a more complete discussion to the full version of the paper.)

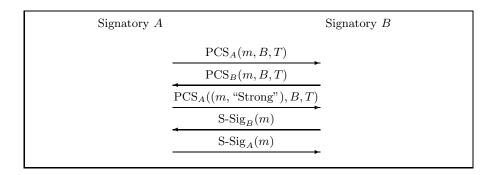


Fig. 2. Alternative abuse-free contract signing protocol for two parties

We now present a contract-signing protocol for three parties that is both abuse-free and fair. The approach we take is to let aborts be overturned, but in a way that avoids any violation of fairness and abuse freeness. The protocol is based on the alternative protocol for two parties shown in Figure 2. In the protocol, the roles of A and B are interchanged during the signature phase: B

sends his converted signature first, and A does it after receiving B's signature. However, B only sends his signature after receiving a new message: a "strong" promise that she won't abort the protocol. One consequence of this is that now T might have to overturn a stored abort outcome, as requested by a cheating A, to that of a valid signed contract. In the full paper we show:

Theorem 4. Assuming the security of DDH, the protocol of Figure 2 is a secure optimistic contract signing protocol for two parties (in the random oracle model).

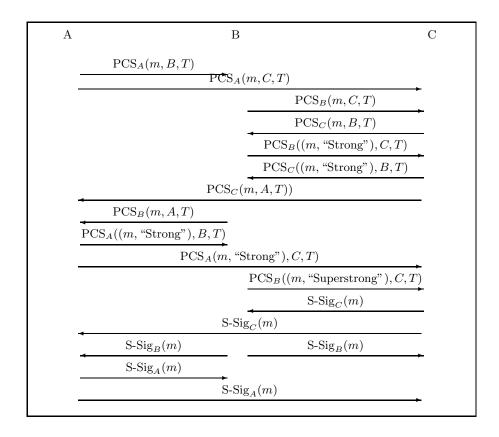


Fig. 3. Secure contract signing protocol for three parties

Based on the alternate protocol for two parties, a secure contract signing protocol for three parties is shown in Figure 3. As in the two-party protocol, the protocol also uses "strong" promises, but now with more levels of strength. In the three-party case, B sends also a "Superstrong" promise not to (support an) abort. Intuitively, and as a general rule, a "Strong" promise presented by a party in a call to resolve will not be used by the trustee to overturn an existing abort;

on the other hand, a "Superstrong" promise will provide overturning power. All the cases are outlined in the description of the protocol, which will be presented in the full version of this paper.

Theorem 5. Assuming the security of DDH, the protocol of Figure 3 is a secure optimistic contract signing protocol for three parties (in the random oracle model).

References

- N. Asokan, B. Baum-Waidner, M. Schunter, and M. Waidner. Optimistic synchronous multi-party contract signing. Technical Report RZ3089, IBM Research Report, 1998. to appear in Verlaessliche Informationssysteme 1999.
- N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In ACM Security'96, pages 6–17.
- N. Asokan, V. Shoup, and M. Waidner. Fair exchange of digital signatures. Technical Report RZ2973, IBM Research Report, 1998. Extended Abstract in Eurocrypt'98.
- B. Baum-Waidner and M. Waidner. Optimistic asynchronous multi-party contract signing. Technical Report RZ3078, IBM Research Report, 1998.
- M. Bellare, J. Garay, and T. Rabin. Fast batch exponentiation for modular exponentiation and digital signatures. In EUROCRYPT'98, pages 236–250.
- M. Bellare and O. Goldreich. On defining proofs of knowledge. In CRYPTO'92, pages 390–420.
- M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Trans. Info. Theory*, 36(1):40–46, 1990.
- 8. M. Blum. Coin flipping by telephone: A protocol for solving impossible problems. In CRYPTO'81, pages 11–15. ECE Report 82-04, 1982.
- M. Blum. How to exchange (secret) keys. ACM Transactions on Computer Systems, 1(2):175–193, May 1983.
- H. Bürk and A. Pfitzmann. Value exchange systems enabling security and unobservability. Computers and Security, 9:715-721, 1990.
- 11. D. Catalano and R. Gennaro. New efficient and secure protocols for verifiable signature sharing and other applications. In CRYPTO'98, pages 105–120.
- 12. D. Chaum. Designated confirmer signatures. In EUROCRYPT'94, pages 86–91.
- D. Chaum and H. V. Antwerpen. Undeniable signatures. In CRYPTO'89, pages 212–216.
- D. Chaum and T. P. Pedersen. Transferred cash grows in size. In EUROCRYPT'92, pages 390–407.
- D. L. Chaum. Silo watching. In CRYPTO'81, pages 138–139. ECE Report 82-04, 1982.
- B. Cox, J. D. Tygar, and M. Sirbu. Netbill security and transaction protocol. In First USENIX Workshop on Electronic Commerce, pages 77–88, 1995.
- 17. R. Cramer. Modular Design of Secure yet Practical Cryptographic Protocols. PhD thesis, University of Amsterdam, 1995.
- 18. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In CRYPTO'94, pages 174–187.
- I. Damgård and T. Pedersen. New convertible undeniable signature schemes. In EUROCRYPT'96, pages 372–386.

- I. B. Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In CRYPTO'89, pages 17–27.
- 21. I. B. Damgård. Practical and provably secure release of a secret and exchange of signatures. *J. of Crypt.*, 8(4):201–222, Autumn 1995.
- 22. R. H. Deng, L. Gong, A. A. Lazar, and W. Wang. Practical protocols for certified electronic mail. *J. of Network and Systems Management*, 4(3), 1996.
- 23. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithm. *IEEE Trans. Info. Theory*, 31:465–472, 1985.
- 24. S. Even. A protocol for signing contracts. ACM SIGACT News, 15(1):34–39, 1983.
- S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. Commun. ACM, 28(6):637–647, June 1985.
- U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In CRYPTO'89, pages 526–545.
- M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed commit with one faulty process. J. ACM, 32(2), 1985.
- 28. M. Franklin and M. Reiter. Fair exchange with a semi-trusted third party. In ACM SECURITY'96, pages 1–5.
- 29. J. Garay and P. MacKenzie. Multi-party contract signing. Manuscript.
- O. Goldreich. A simple protocol for signing contracts. In CRYPTO'83, pages 133–136.
- M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In EUROCRYPT'96, pages 143–154.
- 32. C. H. Lim and P. J. Lee. More flexible exponentiation with precomputation. In CRYPTO'94, pages 95–107.
- S. Micali. Certified e-mail with invisible post offices. Presented at the 1997 RSA Security Conference, 1997.
- 34. M. Michels, H. Petersen and P. Horster. Breaking and repairing a convertible undeniable signature scheme. In ACM SECURITY'96, pages 148–152.
- D. Pointcheval and J. Stern. Security proofs for signature schemes. In EURO-CRYPT'96, pages 387–398.
- C. P. Schnorr. Efficient identification and signatures for smart cards. In CRYPTO'89, pages 239–252.

Can Statistical Zero Knowledge be Made Non-interactive?

or

On the Relationship of \mathcal{SZK} and \mathcal{NISZK}^{\star}

[Extended Abstract]

Oded Goldreich¹, Amit Sahai², and Salil Vadhan²

Department of Computer Science, Weizmann Institute of Science, Rehovot, ISRAEL. E-mail: oded@wisdom.weizmann.ac.il. Work done while visiting LCS, MIT.

² Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: {amits,salil}@theory.lcs.mit.edu. URL: http://theory.lcs.mit.edu/~salil.

Abstract. We extend the study of non-interactive statistical zero-knowledge proofs. Our main focus is to compare the class \mathcal{NISZK} of problems possessing such *non-interactive* proofs to the class \mathcal{SZK} of problems possessing *interactive* statistical zero-knowledge proofs. Along these lines, we first show that if statistical zero knowledge is non-trivial then so is non-interactive statistical zero knowledge, where by non-trivial we mean that the class includes problems which are *not* solvable in probabilistic polynomial-time. (The hypothesis holds under various assumptions, such as the intractability of the Discrete Logarithm Problem.) Furthermore, we show that if \mathcal{NISZK} is closed under complement, then in fact $\mathcal{SZK} = \mathcal{NISZK}$, i.e. all statistical zero-knowledge proofs can be made non-interactive.

The main tools in our analysis are two promise problems that are natural restrictions of promise problems known to be complete for \mathcal{SZK} . We show that these restricted problems are in fact complete for \mathcal{NISZK} and use this relationship to derive our results comparing the two classes. The two problems refer to the statistical difference, and difference in entropy, respectively, of a given distribution from the uniform one. We also consider a weak form of \mathcal{NISZK} , in which only requires that for every inverse polynomial 1/p(n), there exists a simulator which achieves simulator deviation 1/p(n), and show that this weak form of \mathcal{NISZK} actually equals \mathcal{NISZK} .

Keywords: Statistical Zero-Knowledge Proofs, Non-Interactive Zero-Knowledge Proofs.

^{*} The full version of this paper has been submitted to the *Electronic Colloquium on Computational Complexity* [26].

1 Introduction

Zero-Knowledge proofs, introduced by Goldwasser, Micali and Rackoff [29], are fascinating and extremely useful constructs. Their fascinating nature is due to their seemingly contradictory nature; they are both convincing and yet yield nothing beyond the validity of the assertion being proven. Their applicability in the domain of cryptography is vast; they are typically used to force malicious parties to behave according to a predetermined protocol (which requires parties to provide proofs of the correctness of their secret-based actions without revealing these secrets). Zero-knowledge proofs come in many flavors, and in this paper we focus on two parameters: The first parameter is the underlying communication model, and the second is the type of the zero-knowledge quarantee.

The communication model. When Goldwasser, Micali, and Rackoff proposed the definition of zero-knowledge proofs, it seemed that interaction was crucial to achieving zero knowledge – that the possibility of zero knowledge arose through the power of interaction. Indeed, it was not unexpected when [24] showed zero knowledge to be trivial (i.e., only exists for proofs of \mathcal{BPP} statements) in the most straightforward non-interactive models. Surprisingly, however, Blum, Feldman, and Micali [7], showed that by changing the model slightly, it is possible to achieve zero knowledge in a non-interactive setting (i.e. where only unidirectional communication can occur). Specifically, they assume that both Prover and Verifier have access to a shared truly random string, called the reference string. Aside from this assumption, all communication consists of one message, the "proof," which is generated by the Prover (based on the assertion being proved and the reference string) and sent from the Prover to the Verifier.

Non-interactive zero-knowledge proofs, on top of being more communication-efficient by definition, have several applications not offered by ordinary interactive zero-knowledge proofs. They have been used, among other things, to build digital signature schemes secure against adaptive chosen message attack [3], public-key cryptosystems secure against chosen-ciphertext attack [34, 18], and non-malleable cryptosystems [18].

The zero-knowledge guarantee. For ordinary interactive zero-knowledge proofs, the zero-knowledge requirement is formulated by saying that the transcript of the Verifier's interaction with the Prover can be simulated by the Verifier itself. Similarly, for the non-interactive setting described above, the zero-knowledge condition is formulated by requiring that one can produce, knowing only the statement of the assertion, a random reference string along with a "proof" that works for the reference string. More precisely, we require that there exists an efficient procedure that on input a valid assertion produces a distribution which is "similar" to the joint distribution of random reference strings and proofs generated by the Prover. The key parameter is the interpretation of "similarity." Two notions have been commonly considered in the literature (cf., [29, 23, 21, 6, 5]). Statistical zero knowledge requires that these distributions be statistically close (i.e., the statistical difference between them is negligible). Computational zero

knowledge instead requires that these distributions are computationally indistinguishable (cf., [28, 41]). In this work, we focus on the stronger security requirement of statistical zero knowledge.

Since its introduction in [7], most work on non-interactive zero knowledge has focused on the computational type (cf., [7, 15, 16, 6, 20, 31]). With non-interactive statistical zero knowledge, the main objects of investigation have been the specific proof system for Quadratic Nonresiduosity and variants [6, 14, 11]. Recently, De Santis *et. al.* [12] opened the door to a general study of non-interactive statistical zero-knowledge by showing that it contains a complete (promise²) problem.

Notation. Throughout the paper, \mathcal{SZK} denotes the class of promise problems having statistical zero-knowledge interactive proof systems (defined in Appendix A), and \mathcal{NISZK} denotes the class of promise problems having non-interactive statistical zero-knowledge proof systems (defined in Section 1.1).

Our Contribution. In this work, we seek to understand what, if any, additional power interaction gives in the context of statistical zero knowledge. Thus, we continue the investigation of \mathcal{NISZK} , focusing on its relationship with \mathcal{SZK} . Our first result is that the non-triviality of \mathcal{SZK} implies non-triviality of \mathcal{NISZK} , where by non-trivial we mean that a class includes problems which are *not* solvable in probabilistic polynomial-time. The hypothesis holds under various assumptions, such as the intractability of Discrete Logarithm Problem [22] (or Quadratic Residuosity [29] or Graph Isomorphism [23]), but variants of these last two problems are already known to be in \mathcal{NISZK} [6, 5]).

Furthermore, we show that if \mathcal{NISZK} is closed under complement, then in fact $\mathcal{SZK} = \mathcal{NISZK}$ — i.e., all statistical zero-knowledge proofs can be made non-interactive. We note that [12] have claimed that \mathcal{NISZK} is closed under complement (and OR), but these claims have been retracted [13].

We also show the equivalence of \mathcal{NISZK} with a variant in which the statistical zero knowledge requirement is weakened somewhat.

Complete Problems. Central to our methodology is the use of simple and natural complete problems to understand classes, such as \mathcal{SZK} and \mathcal{NISZK} , whose definitions are rather complicated. In particular, we exhibit two natural promise problems and prove that they are complete for \mathcal{NISZK} . The two problems refer to the "distance" (in two different senses) of a given distribution from the uniform one. These two problems are natural restrictions of two promise problems shown complete for \mathcal{SZK} , in [38] and [27], respectively. Indeed, our results about the

¹ The only exception is an unpublished manuscript of Bellare and Rogaway [5] who proved some basic results about non-interactive perfect zero-knowledge and showed a non-interactive perfect zero-knowledge proof for the language of graphs with trivial automorphism group.

² A promise problem Π is a pair $\Pi = (\Pi_{YES}, \Pi_{NO})$ of disjoint sets of strings, corresponding to YES and NO instances of a decision problem.

relationship between \mathcal{SZK} and \mathcal{NISZK} come from relating the corresponding complete problems. This general theme of using completeness to simplify the study of a class, rather than as evidence for computational intractability (as is the traditional use of \mathcal{NP} -completeness), has been evidenced in a number of recent works (cf., [23, 33, 40, 1, 2]) and has been particularly useful in understanding statistical zero knowledge (cf., [38, 39, 12, 27]).

1.1 The Non-interactive Model

Let us recall the definition of a non-interactive statistical zero-knowledge proof system from [6].³ We will adapt the definition to promise problems. Note that our definition will capture what [6] call a bounded proof system, in that each shared reference string can only be used once. In contrast to non-interactive computational zero knowledge (cf., [6, 20]), it is unknown whether every problem that has such a (bounded) non-interactive statistical zero-knowledge proof system also has one in which the shared reference string can be used an unbounded (polynomial) number of times.

A non-interactive statistical zero-knowledge proof system for a promise problem Π is defined by a triple of probabilistic machines P, V, and S, where V and S are polynomial-time and P is computationally unbounded, and a polynomial r(n) (which will give the size of the random reference string σ), such that:

- 1. (Completeness:) For all $x \in \Pi_{YES}$, the probability that $V(x, \sigma, P(x, \sigma))$ accepts is at least 2/3.
- 2. (Soundness:) For all $x \in \Pi_{NO}$, the probability that $V(x, \sigma, P(x, \sigma))$ accepts is at most 1/3.
- 3. (Zero Knowledge:) For all $x \in \Pi_{YES}$, the statistical deviation between the following two distributions is at most $\beta(|x|)$:
 - (A) Choose σ uniformly from $\{0,1\}^{r(|x|)}$, sample p from $P(x,\sigma)$, and output (p,σ) .
 - (B) S(x) (where the coins for S are chosen uniformly at random.)

where $\beta(n)$ is a negligible function,⁴ termed the *simulator deviation*, and the probabilities in Conditions 1 and 2 are taken over the random coins of V and P, and the choice of σ uniformly from $\{0,1\}^{r(n)}$. Note that non-interactive statistical zero knowledge is closed under parallel repetition, so the completeness and soundness errors (i.e. the probability of rejection (resp., acceptance) for YES (resp., NO) instances) can be made exponentially small in |x|.

We also define a weaker notion of zero knowledge, known as a weak non-interactive statistical zero-knowledge proof system, where we ask only that for every polynomial g(n), there exists a probabilistic polynomial-time simulator

³ Actually, only non-interactive *perfect* and computational zero-knowledge proofs were defined in [6]. The definition we are using, previously given in [5, 12], is the natural non-interactive analogue of (interactive) statistical zero knowledge [29].

⁴ Recall that a function is *negligible* if it is eventually less than 1/g(n) for any polynomial g.

 S_g (whose running time may depend on g), such that the simulator deviation as defined above is at most 1/g(|x|). This is the natural analogue of a notion defined in the interactive setting for statistical zero knowledge [17] as well as concurrent zero knowledge [19].

The class of promise problems that possess non-interactive statistical zero-knowledge proof systems is denoted \mathcal{NISZK} , and we denote by $weak\text{-}\mathcal{NISZK}$ the class of promise problems that possess weak non-interactive statistical zero-knowledge proof systems. Note that by definition, $\mathcal{NISZK} \subset weak\text{-}\mathcal{NISZK}$. De Santis $et.\ al.\ [12]$ recently began a general investigation of the class \mathcal{NISZK} . They introduced a promise problem, called Image Density, and claimed that is complete for \mathcal{NISZK} and that the latter class is closed under OR and complement. We were able to verify that some variants of Image Density are \mathcal{NISZK} -complete, and indeed the ideas used towards this goal are important to our work. However, they have retracted their claims that \mathcal{NISZK} is closed under OR and complement [13].

In this paper, in addition to examining \mathcal{NISZK} on its own, we also consider the relationship non-interactive statistical zero-knowledge proofs have with interactive statistical zero-knowledge proofs. In the context of interactive zero-knowledge proofs, another issue that arises in the zero-knowledge condition is the behavior of the verifier. The general definition of zero knowledge requires that the zero-knowledge requirement hold for any probabilistic polynomial-time verifier. A weaker requirement, called honest-verifier zero knowledge, requires the zero-knowledge condition to hold only if the verifier behaves honestly. However, it is known that these two conditions are equivalent for statistical zero knowledge, in the sense that every statistical zero-knowledge proof against the honest verifier can be transformed into one that is statistical zero knowledge against any verifier [25]. Thus, we write \mathcal{SZK} for the class of promise problems possessing statistical zero-knowledge proofs (against any polynomial-time verifier or, equivalently, against just the honest verifier).

Note that in the case of non-interactive zero knowledge, the issue of honest verifiers does not arise since the verifier does not interact with the prover. Also, note that we can always transform a non-interactive zero-knowledge proof into an honest verifier zero-knowledge proof, since we could have the honest verifier supply a random string which can replace the common reference string required for non-interactive zero knowledge. That is, $\mathcal{NISZK} \subset \mathcal{SZK}$ (recalling the equivalence of \mathcal{SZK} with honest-verifier \mathcal{SZK}).

1.2 Our Results

The primary tools we use in our investigation are promise problems that are complete for \mathcal{SZK} or \mathcal{NISZK} . In [38], a promise problem called Statistical Difference (SD) was introduced and proved complete for \mathcal{SZK} , providing the first completeness result for \mathcal{SZK} . Recently, it was shown in [27] that another natural problem, called Entropy Difference (ED), is complete for \mathcal{SZK} as well. In this work, we show that "one-sided" versions of these problems, which we call Statistical Difference from Uniform (SDU) and Entropy Approximation (EA), are

complete for \mathcal{NISZK} . To define these problems more precisely, we first recall that that statistical difference between two random variables X and Y on a finite set D, denoted $\Delta(X, Y)$, is defined to be

$$\Delta(X, Y) \stackrel{\text{def}}{=} \max_{S \subset D} |\Pr[X \in S] - \Pr[Y \in S]| = \frac{1}{2} \cdot \sum_{\alpha} |\Pr[X = \alpha] - \Pr[Y = \alpha]|.$$

All the promise problems we consider involve distributions which are encoded by circuits which sample from them. That is, if X is a circuit mapping $\{0,1\}^m$ to $\{0,1\}^n$, we identify X with the probability distribution induced on $\{0,1\}^n$ by feeding X the uniform distribution on $\{0,1\}^m$. Since circuits can be evaluated in time polynomial in their size, yet are rich enough to encode general (e.g., Turing machine) computations, they effectively capture the notion of an "efficiently sampleable distribution."

Definition 1.1. (Problems involving statistical difference): The promise problem Statistical Difference, denoted $SD = (SD_{YES}, SD_{NO})$, consists of

$$\begin{split} & \operatorname{SD}_{\mathrm{YES}} \stackrel{\mathrm{def}}{=} \{(X,Y) : \varDelta(X\,,\,Y) < 1/3 \} \\ & \operatorname{SD}_{\mathrm{NO}} \stackrel{\mathrm{def}}{=} \{(X,Y) : \varDelta(X\,,\,Y) > 2/3 \} \end{split}$$

where X and Y are distributions encoded as circuits which sample from them. Statistical Difference from Uniform, denoted SDU = (SDU_{YES}, SDU_{NO}), consists of

$$\begin{split} & \mathtt{SDU}_{\mathrm{YES}} \stackrel{\mathrm{def}}{=} \{X: \Delta(X\,,\,U) < 1/n\} \\ & \mathtt{SDU}_{\mathrm{NO}} \stackrel{\mathrm{def}}{=} \{X: \Delta(X\,,\,U) > 1 - 1/n\} \end{split}$$

where X is a distribution encoded as a circuit outputting n bits, and U is the uniform distribution on n bits.

For the two problems related to entropy, we recall that the (Shannon) entropy of a random variable X, denoted H(X), is defined as

$$\mathrm{H}(X) \stackrel{\mathrm{def}}{=} \sum_{\alpha} \Pr\left[X = \alpha\right] \cdot \log_2(1/\Pr\left[X = \alpha\right])$$

Definition 1.2. (Problems involving entropy): The promise problem Entropy Difference, $denoted \ ED = (ED_{YES}, ED_{NO}), \ consists \ of$

$$\begin{split} & \mathtt{ED}_{\mathrm{YES}} \stackrel{\mathrm{def}}{=} \{(X,Y) : \mathtt{H}(X) > \mathtt{H}(Y) + 1\} \\ & \mathtt{ED}_{\mathrm{NO}} \stackrel{\mathrm{def}}{=} \{(X,Y) : \mathtt{H}(Y) > \mathtt{H}(X) + 1\} \end{split}$$

Entropy Approximation, $denoted EA = (EA_{YES}, EA_{NO}), consists of$

$$\begin{split} \mathtt{EA}_{\mathrm{YES}} &\stackrel{\mathrm{def}}{=} \{(X,k) : \mathrm{H}(X) > k+1\} \\ \mathtt{EA}_{\mathrm{NO}} &\stackrel{\mathrm{def}}{=} \{(X,k) : \mathrm{H}(X) < k-1\} \end{split}$$

In these problems, k is a positive integer and X and Y are distributions encoded as circuits which sample from them.

Our first theorem, which is the starting point for our other results, is:

Theorem 1.3. (EA and SDU are \mathcal{NISZK} -complete) The promise problems EA and SDU are complete for \mathcal{NISZK} . That is, EA, SDU $\in \mathcal{NISZK}$ and for every promise problem $\Pi \in \mathcal{NISZK}$, there is a polynomial-time Karp (many-one) reduction from Π to EA and another from Π to SDU.

From the proof of this theorem, we also obtain a method for transforming weak non-interactive statistical zero knowledge proofs into standard ones.

Theorem 1.4. weak- $\mathcal{NISZK} = \mathcal{NISZK}$.

Armed with our complete problems, we then begin the work of comparing \mathcal{SZK} and \mathcal{NISZK} . First we show that the non-triviality of \mathcal{NISZK} is equivalant to the non-triviality of \mathcal{SZK} . This is shown by giving a Cook reduction from ED to EA.

Theorem 1.5. (non-triviality of \mathcal{NISZK}) $\mathcal{SZK} \neq \mathcal{BPP} \iff \mathcal{NISZK} \neq \mathcal{BPP}$.

In this theorem (and throughout the paper), \mathcal{BPP} denotes the class of *promise* problems solvable in probabilistic polynomial time.

In fact, it turns out that the type of Cook reduction we use is a special one, and by examining it further, we are able to shed more light on the \mathcal{SZK} vs. \mathcal{NISZK} question. Specifically, we observe that the reduction we give from ED to EA is an \mathcal{AC}^0 truth-table reduction. That is, it is a nonadaptive Cook reduction in which the postprocessing is done in \mathcal{AC}^0 . (Formal definitions are given in Section 5.2.) Further, we can prove that if \mathcal{NISZK} is closed under complement, then \mathcal{NISZK} is closed under \mathcal{AC}^0 truth-table reductions. Thus we deduce that \mathcal{NISZK} being closed under complement implies that $\mathcal{NISZK} = \mathcal{SZK}$. In fact, we can show that closure under complement and a number of other natural conditions are equivalent to $\mathcal{SZK} = \mathcal{NISZK}$:

Theorem 1.6. (conditions for SZK = NISZK) The following are equivalent:

- 1. SZK = NISZK.
- 2. NISZK is closed under complement.
- 3. \mathcal{NISZK} is closed under \mathcal{NC}^{1} truth-table reductions.
- 4. ED (resp., SD) Karp-reduces to EA (resp., SDU). ("general versions reduce to one-sided ones")
- 5. EA (resp., SDU) Karp-reduces to its complement. ("one-sided versions reduce to their complements")

Theorem 1.6 can be interpreted as saying that if \mathcal{NISZK} has a relatively weak closure property (closure under complement), then the class is surprisingly rich (equals \mathcal{SZK}) and has a much stronger closure property (closure under \mathcal{NC}^1 truth-table reductions.) At first, it might seem implausible that a class like \mathcal{NISZK} with such an assymetric definition would be closed under complement. But \mathcal{SZK} , which has a similarly assymetric definition, is known to be closed

under complement [35]. In light of this, the closure of \mathcal{NISZK} under complement would not be quite as unexpected, and Theorem 1.6 illustrates that proving it would have wider consequences.

The last two conditions in Theorem 1.6 show that these questions about non-interactive versus interactive statistical zero-knowledge proofs are actually equivalent to basic questions about relationships between natural computational problems whose definitions have no *a priori* relationship to zero-knowledge proofs.

The equality of \mathcal{SZK} and \mathcal{NISZK} has interesting consequences not just for \mathcal{NISZK} , but also for \mathcal{SZK} . Currently, the best known generic protocol for \mathcal{SZK} (against cheating verifiers, making no computational assumptions) requires a polynomial number of rounds [35, 25].⁵ For \mathcal{NISZK} , however, by [10], it is known that every problem in \mathcal{NISZK} has a constant round statistical zero-knowledge proof system (against general, cheating verifiers) with inverse polynomial soundness error. Whether every problem in \mathcal{SZK} has such a proof system is still an open question, which would be resolved in the positive if $\mathcal{SZK} = \mathcal{NISZK}$.

1.3 A Wider Perspective

The study of non-interactive statistical (rather than computational) zero-knowledge proofs may be of interest for two reasons. Firstly, statistical zero-knowledge proofs provide an almost absolute level of security, whereas computational zeroknowledge proofs only provide security relative to computational abilities (and typically under complexity theoretic assumptions). Secondly, by analogy from the study of zero-knowledge interactive proofs, we believe that techniques developed for the "cleaner" statistical model can be applied or augmented to yield results for computational zero-knowledge: The proof that one-way functions are necessary for \mathcal{SZK} to be non-trivial [36] was later generalized to \mathcal{CZK} [37]. More recently, the transformations of honest-verifier zero knowledge to general zero knowledge, presented in [8, 10, 9, 25], apply both to statistical and computational zero knowledge (whereas the original motivation was the study of statistical zero knowledge). It is our hope that the current study of \mathcal{NISZK} will eventually lead to a better understanding of \mathcal{NICZK} , where there are still important open questions such as the minimal conditions under which \mathcal{NP} has \mathcal{NICZK} proofs.

2 Preliminaries

Recall that a promise problem Π is a pair $\Pi = (\Pi_{YES}, \Pi_{NO})$ of disjoint sets of strings, corresponding to the following decision problem: Given a string $x \in \Pi_{YES} \cup \Pi_{NO}$, decide whether it is in Π_{YES} (i.e. is a YES instance) or in Π_{NO} (i.e. is a NO instance). A string in $\Pi_{YES} \cup \Pi_{NO}$ is said to satisfy the promise, and all

⁵ Under the assumption that the Discrete Logarithm is hard, however, there is a constant round, cheating verifier SZK proof system with inverse polynomial soundness error for all of SZK [35, 4].

other strings are said to violate the promise. A function f is said to be a Karp (or polynomial-time many-one) reduction from a promise problem Π to a promise problem Γ if f is polynomial-time computable, $x \in \Pi_{YES} \Rightarrow f(x) \in \Gamma_{YES}$, and $x \in \Pi_{NO} \Rightarrow f(x) \in \Gamma_{NO}$. If such a reduction exists, we write $\Pi \leq_{Karp} \Gamma$.

Recall that all the promise problems we are considering involve distributions which are encoded by circuits which sample from them. That is, if X is a circuit mapping $\{0,1\}^m$ to $\{0,1\}^n$, we identify X with the probability distribution induced on $\{0,1\}^m$ by feeding X the uniform distribution on $\{0,1\}^m$. The support of X is the set of strings in $\{0,1\}^n$ which have nonzero probability under X, i.e. $\{y \in \{0,1\}^n : \exists r \in \{0,1\}^m \text{ s.t. } X(r) = y\}$. For any distribution X on a set D, we write $\otimes^k X$ to denote the distribution on D^k consisting of k independent copies of X.

3 EA is in \mathcal{NISZK}

In this section, we show that EA has a non-interactive statistical zero-knowledge proof system.

Lemma 3.1. EA $\in \mathcal{NISZK}$. Moreover, there is a non-interactive statistical zero-knowledge proof system for EA in which the completeness error, soundness error, and simulator deviation are all exponentially vanishing (specifically 2^{-s} , where s is the length of the input).

The transformation given by the following lemma will be applied at the start of the proof system:

Lemma 3.2. There is a polynomial-time computable function that takes an instance (X, k) of EA and a parameter s (in unary) and produces a distribution Z on $\{0,1\}^{\ell}$ (encoded by a circuit which samples from it) such that

- 1. If H(X) > k+1, then Z has statistical difference at most 2^{-s} from the uniform distribution on $\{0,1\}^{\ell}$, and
- 2. If H(X) < k-1, then the support of Z is at most a 2^{-s} fraction of $\{0,1\}^{\ell}$.

The proof of Lemma 3.2 uses 2-universal hashing and the Leftover Hash Lemma [30] and is the most technically involved part of this work. However, due to space constraints, the construction and proof are deferred to the full version of the paper [26]. Lemma 3.2 essentially transforms an instance of Entropy Approximation into an instance of Image Density, the complete problem of [12]. Given this transformation, it is straightforward to give a noninteractive statistical zero-knowledge proof system for EA:

Non-interactive proof system for EA, on input (X, k)

- 1. Let Z be the distribution on $\{0,1\}^{\ell}$ obtained from (X,k) as in Lemma 3.2 taking s to be the total description length of (X,k) in bits. Let $\sigma \in \{0,1\}^{\ell}$ be the reference string.
- 2. P selects r uniformly among $\{r': Z(r') = \sigma\}$ and sends r to V.

3. V accept if $Z(r) = \sigma$ and rejects otherwise.

It is immediate from Lemma 3.2 that the completeness error and soundness error of this proof system are 2^{-s} . For zero-knowledgeness, we consider the following probabilistic polynomial-time simulator:

Simulator for EA proof system, on input (X, k)

- 1. Let Z be obtained from (X, k) as in the proof system.
- 2. Select an input r to Z uniformly at random and let $\sigma = Z(r)$.
- 3. Output (σ, r) .

It follows from Part 1 of Lemma 3.2 that this simulator has statistical difference at most 2^{-s} from the distribution of transcripts of (P, V). Thus, assuming Lemma 3.2, we have established Lemma 3.1. In fact, we need not require that s be the length of (X, k). Instead, s can be taken to be an arbitrary security parameter, and the completeness, soundness, and simulation error will be exponentially small in s, while the running time of the protocol only depends polynomially on s. We can use this to prove the following, which will be useful to us later.

Proposition 3.3. If any promise problem Π reduces to EA by a Karp (i.e. manyone) reduction (even if it is length-reducing), then $\Pi \in \mathcal{NISZK}$.

Proof. A noninteractive statistical zero-knowledge proof system for Π can be given as follows: On an instance x of Π , both parties compute the image (X,k) of x under the reduction $\Pi \leq_{\texttt{Karp}} \texttt{EA}$ and execute the proof system for EA on (X,k), except that we take s to be the length of x. Hence, the completeness and soundness errors and simulator deviation of this proof system are exponentially small in |x| (rather than |(X,k)| which could be shorter than x).

4 EA and SDU are \mathcal{NISZK} -complete

In this section, we complete the proof of Theorem 1.3. First, we establish that $\mathtt{SDU} \in \mathcal{NISZK}$ by showing:

Lemma 4.1. $SDU \leq_{Karp} EA$. In particular, $SDU \in \mathcal{NISZK}$.

Proof. Let X be an instance of SDU. We assume that $\log(n) > 5$, where n is the output length of the circuit X (otherwise, once can decide in probabilistic polynomial time whether X is a YES or NO instance of SDU by random sampling). Let U denote the uniform distribution on n bits. We claim the map $X \mapsto (X, n-3)$ is the reduction required by the lemma.

If $X \in SDU_{YES}$, then $\delta = \Delta(X, U) < 1/n$. Now we use the fact (see, e.g., [27]) that for any two random variables, Y and Z, ranging over domain D it holds that

$$|\mathrm{H}(Y) - \mathrm{H}(Z)| \leq (\log |D|) \cdot \Delta(Y, Z) + \mathrm{H}_2(\Delta(Y, Z)),$$

where $H_2(\theta)$ denotes the entropy of a 0–1 random variable with mean θ . Applying this with Y = U and Z = X, we have

$$n - H(X) < n \cdot 1/n + H_2(1/n) < 2.$$

Hence $(X, n-3) \in EA_{YES}$.

If $X \in SDU_{NO}$, then $\Delta(X, U) \geq 1 - 1/n$. By the definiton of statistical difference, this implies the existence of a set $S \subset \{0, 1\}^n$ such that $\Pr[X \in S] - \Pr[U \in S] > 1 - 1/n$. This implies that

$$\Pr[X \in S] > 1 - 1/n$$
 and $\Pr[U \in S] < 1/n$.

Thus, $H(X) \leq \Pr[X \in S] \cdot \log(|S|) + \Pr[X \notin S] \cdot n < 1 \cdot (n - \log n) + (1/n) \cdot n < n - 4$, and we have that $(X, n - 3) \in EA_{NO}$.

The "in particular" part of Lemma 4.1 follows immediately from Proposition 3.3.

Now, we establish both Theorem 1.3 and Theorem 1.4 by showing that all promise problems in weak- \mathcal{NISZK} (and hence all promise problems in \mathcal{NISZK}) are reducible to SDU (and hence by the previous lemma to EA).

Lemma 4.2. Every promise problem in weak-NISZK Karp-reduces to SDU.

Proof. Let Π be any promise problem in weak-NISZK. As weak-NISZK is preserved under parallel repetition, we may assume that Π has a weak-NISZK proof system (P, V) with completeness and soundness errors at most 2^{-n} on inputs of length n. Let r(n) = poly(n) be the length of the random reference string in (P, V), and let S be a randomized polynomial-time simulator S such that the statistical difference between the output distribution of S and the distribution of true transcripts of P is at most 1/(3r(n)). (Such an S is guaranteed by the weak-NISZK property.) Let U denote the uniform distribution on r(n) bits.

Let x be an instance of Π . Define M_x to be a circuit which does the following on input s:

 $M_x(s)$: Simulate S(x) with randomness s to obtain a transcript (σ, p) . If $V(x, \sigma, p)$ accepts, then output σ , else output $0^{r(n)}$.

We claim that the map $x \mapsto M_x$ is the reduction required by the lemma. Suppose $x \in \Pi_{YES}$. In this case, we know that the random reference string σ in the output of S has statistical difference less than 1/3r(n) from U. In addition, since the completeness error of protocol P is at most 2^{-n} , S(x) can output rejecting transcripts with probability at most $1/(3r(n)) + 2^{-n} \leq 2/(3r(n))$. Hence, $\Delta(M_x, U) < 2/(3r(n)) + 1/(3r(n)) \leq 1/r(n)$, and $M_x \in SDU_{YES}$.

Suppose $x \in \Pi_{\text{NO}}$. Since the soundness error of protocol P is bounded by 2^{-n} , for at most a 2^{-n} fraction of reference strings σ does there exist an accepting transcript (σ, p) . Since M_x only outputs reference strings corresponding to accepting transcripts or $0^{r(n)}$, $\Delta(M_x, U) \geq 1 - (2^{-n} + 2^{-r(n)}) > 1 - 1/r(n)$. Thus, $M_x \in \text{SDU}_{\text{NO}}$.

Clearly, Lemmas 3.1, 4.1, and 4.2 combine to prove Theorem 1.3. Lemmas 4.2 and 4.1 show that any promise problem Π in weak- \mathcal{NISZK} reduces to EA; by Proposition 3.3, this implies that $\Pi \in \mathcal{NISZK}$ and establishes Theorem 1.4.

5 Comparing \mathcal{NISZK} and \mathcal{SZK}

Armed with \mathcal{NISZK} -complete promise problems so closely related to problems known to be complete for \mathcal{SZK} , we can quickly begin relating the two classes.

5.1 Nontriviality of \mathcal{NISZK}

First, we establish Theorem 1.5 by giving a Cook reduction from Entropy Difference (ED), complete for \mathcal{SZK} , to Entropy Approximation (EA), complete for \mathcal{NISZK} .

Lemma 5.1. Suppose (X,Y) is an instance of ED. Let $X' = \otimes^4 X$ (resp., $Y' = \otimes^4 Y$) consist of 4 independent copies of X (resp., Y), and let n denote the maximum of the output sizes of X' and Y'. Then,

$$\begin{split} (X,Y) \in \mathrm{ED}_{\mathrm{YES}} &\Longrightarrow \bigvee_{k=1}^{n} \left[\left((X',k) \in \mathrm{EA}_{\mathrm{YES}} \right) \wedge \left((Y',k) \in \mathrm{EA}_{\mathrm{NO}} \right) \right] \\ (X,Y) \in \mathrm{ED}_{\mathrm{NO}} &\Longrightarrow \bigwedge_{k=1}^{n} \left[\left((X',k) \in \mathrm{EA}_{\mathrm{NO}} \right) \vee \left((Y',k) \in \mathrm{EA}_{\mathrm{YES}} \right) \right] \end{split}$$

Proof. Suppose $(X,Y) \in \mathtt{ED}_{YES}$, so that H(X') > H(Y') + 4. Let $k = \lfloor H(X') \rfloor - 2$. Then H(X') > k+1. On the other hand, k+3 > H(X') > H(Y') + 4, and hence H(Y') < k-1. Suppose instead $(X,Y) \in \mathtt{ED}_{NO}$, so that H(Y') > H(X') + 4. Then for all $k > \lceil H(X') \rceil + 1$, we have H(X') < k-1. So, for all $k \leq \lceil H(X') \rceil + 1$, we have k+1 < H(X') + 3 < H(Y').

From this reduction, we conclude that $\mathcal{SZK} \neq \mathcal{BPP} \iff \mathcal{NISZK} \neq \mathcal{BPP}$, which is Theorem 1.5. Again, by \mathcal{BPP} we mean the class of *promise problems* solvable in probabilistic polynomial time.

Proof of Theorem 1.5. By definition, $\mathcal{NISZK} \subset \mathcal{SZK}$ (recall that \mathcal{SZK} equals honest-verifier \mathcal{SZK} [25]). Hence if $\mathcal{SZK} = \mathcal{BPP}$, then $\mathcal{NISZK} = \mathcal{BPP}$.

Now suppose $\mathcal{NISZK} = \mathcal{BPP}$, so in particular there is a probabilistic polynomial-time machine M which decides EA (with exponentially small error probability). To show $\mathcal{SZK} = \mathcal{BPP}$, it suffices to show that $\mathsf{ED} \in \mathcal{BPP}$ since ED is \mathcal{SZK} -complete. We now describe how to decide instances of ED : Let (X,Y) be an instance of ED . Letting X' and Y' be as stated in Lemma 5.1, we run M(X',k) and M(Y',k) for all $k \in [1,n]$. If for some k, we see that M(X',k) = 1 and M(Y',k) = 0, we output 1. Otherwise, we output 0. By Lemma 5.1, this is a correct \mathcal{BPP} algorithm for deciding ED . \square

5.2 Conditions under which $\mathcal{NISZK} = \mathcal{SZK}$

The reduction given by Lemma 5.1 is a very special type of Cook reduction, which we call an \mathcal{AC}^0 truth-table reduction. In this section, we use the special properties of this reduction to show that if \mathcal{NISZK} is closed under complement, then in fact $\mathcal{NISZK} = \mathcal{SZK}$. We now precisely define the types of reductions we are using, taking care how they are defined for promise problems.

Definition 5.2. (truth-table reduction [32]): We say a promise problem Π truth-table reduces to a promise problem Γ , written $\Pi \leq_{\mathsf{tt}} \Gamma$, if there exists a (deterministic) polynomial-time computable function f, which on input x produces a tuple (x_1, x_2, \ldots, x_k) and a circuit C, such that

- 1. If $x \in \Pi_{YES}$ then for all valid settings of b_1, b_2, \ldots, b_k , $C(b_1, b_2, \ldots, b_k) = 1$, and
- 2. If $x \in \Pi_{NO}$ then for all valid settings of b_1, b_2, \dots, b_k , $C(b_1, b_2, \dots, b_k) = 0$.

where a setting for b_i is considered valid when $b_i = 1$ if $x_i \in \Gamma_{YES}$ and $b_i = 0$ if $x_i \in \Gamma_{NO}$ (and b_i is unrestricted when x_i violates the promise).

In other words, a truth-table reduction for promise problems is a non-adaptive Cook reduction which is allowed to make queries which violate the promise, but must be able to tolerate both yes and no answers in response to queries that violate the promise. We further consider the case where we restrict the complexity of computing the output of the reduction from the queries:

Definition 5.3. $(\mathcal{AC}^0 \text{ and } \mathcal{NC}^1 \text{ truth-table reductions})$: A truth-table reduction f between promise problems is an \mathcal{AC}^0 (resp., \mathcal{NC}^1) truth-table reduction if the circuit C produced by the reduction on input x has depth bounded by a constant c_f independent of x (resp., has depth bounded by $c_f \log |x|$). If there is an \mathcal{AC}^0 (resp., \mathcal{NC}^1) truth-table reduction from Π to Γ , we write $\Pi \leq_{\mathcal{AC}^0 - \mathsf{tt}} \Gamma$ (resp., $\Pi \leq_{\mathcal{NC}^1 - \mathsf{tt}} \Gamma$).

With this definition, we observe that Lemma 5.1 in fact gives an \mathcal{AC}^0 truthtable reduction, since the formula given in the lemma can be expressed as an \mathcal{AC}^0 circuit, and the statement of the lemma shows that the reduction has the robustness properties against promise violations that are required in Definition 5.3. Thus, we have:

Proposition 5.4. $ED \leq_{AC^{\circ}-tt} EA$.

We say that a class \mathcal{C} of promise problems is closed under a class of reductions \leq_* if $\Pi \leq_* \Gamma$ and $\Gamma \in \mathcal{C}$ implies that $\Pi \in \mathcal{C}$. By the above, if \mathcal{NISZK} is closed under \mathcal{AC}^0 truth-table reductions, then $\mathrm{ED} \in \mathcal{NISZK}$ and hence $\mathcal{NISZK} = \mathcal{SZK}$. Thus, we would like to capture the minimal conditions necessary for a promise class to be closed under \mathcal{AC}^0 truth-table reductions. Here, care must be taken to because of the possibility of promise violations. Keeping this in mind, we define the following operator on promise problems to capture the notion of an unbounded fan-in AND gate for promise problems:

Definition 5.5. (unbounded AND): For any promise problem Π , we define AND(Π) to be the promise problem:

$$\begin{split} & \text{AND}_{\text{YES}}(\Pi) \stackrel{\text{def}}{=} \{(x_1, x_2, \dots, x_k) : k \geq 0, \forall i \in [1, k] x_i \in \Pi_{\text{YES}} \} \\ & \text{AND}_{\text{NO}}(\Pi) \stackrel{\text{def}}{=} \{(x_1, x_2, \dots, x_k) : k \geq 0, \exists i \in [1, k] x_i \in \Pi_{\text{NO}} \} \end{split}$$

We say a class of promise problems $\mathcal C$ is closed under unbounded AND if $\Pi \in \mathcal C$ implies that $\mathtt{AND}(\Pi) \in \mathcal C$.

480

We have defined AND so that it has the weakest promise condition possible to remain well-defined. In particular, we see that $AND_{NO}(\Pi)$ is defined to include x_i 's that violate Π 's promise, as long as just *one* of them is in Π_{NO} . $\Pi \in \mathcal{C}$, $AND(\Pi) \in \mathcal{C}$. We also need a way of combining two promise problems:

Definition 5.6. (disjoint union): For any pair of promise problems Π and Γ , we define the disjoint union of Π and Γ to be the promise problem $\mathtt{DisjUn}(\Pi,\Gamma)$ defined as follows:

$$\begin{split} \operatorname{\mathtt{DisjUn}}_{\mathrm{YES}}(\Pi, \varGamma) &\stackrel{\mathrm{def}}{=} \{0\} \times \varPi_{\mathrm{YES}} \cup \{1\} \times \varGamma_{\mathrm{YES}} \\ \operatorname{\mathtt{DisjUn}}_{\mathrm{NO}}(\Pi, \varGamma) &\stackrel{\mathrm{def}}{=} \{0\} \times \varPi_{\mathrm{NO}} \cup \{1\} \times \varGamma_{\mathrm{NO}} \end{split}$$

We say a class of promise problems $\mathcal C$ is closed under disjoint union if $\Pi, \Gamma \in \mathcal C$ implies that $\mathrm{DisjUn}(\Pi, \Gamma) \in \mathcal C$.

With these definitions, we can give the following lemma which gives some conditions sufficient to give closure under AC^0 truth-table reductions.

Lemma 5.7. A promise class C is closed under AC^0 truth-table reductions if the following conditions hold:

- 1. C is closed under Karp (i.e., many-one) reductions.
- 2. C is closed under unbounded AND.
- 3. C is closed under disjoint union.
- 4. C is closed under complementation.

Lemma 5.7 can be proven by a straightforward induction on the depth of the circuits. Details are given in the full version of the paper [26]. Which of the conditions of Lemma 5.7 does \mathcal{NISZK} satisfy? We argue that Conditions 1, 2, and 3 are satisfied by \mathcal{NISZK} : Closure under Karp reductions and disjoint union follows readily from Proposition 3.3 and the completeness of EA. For closure under unbounded AND, note that to give an \mathcal{NISZK} proof for the AND of many statements, one can give individual \mathcal{NISZK} proofs for each of the statements in parallel. The only technical difficulty is that the lengths of the statements are not guaranteed to be polynomially related, but this can be dealt with as in the proof of Proposition 3.3 or by noting that instances of EA can be trivially padded. Thus, we have the following lemmas (whose full proofs are given in the full version of this paper [26]):

Lemma 5.8. \mathcal{NISZK} is closed under Karp reductions.

Lemma 5.9. \mathcal{NISZK} is closed under unbounded AND.

Lemma 5.10. \mathcal{NISZK} is closed under disjoint union.

Combining everything, we can give a condition under which $\mathcal{SZK} = \mathcal{NISZK}$.

Proposition 5.11. If NISZK is closed under complementation, then SZK = NISZK.

Proof. Suppose \mathcal{NISZK} is closed under complementation. Combining this with Lemmas 5.7, 5.8, 5.9, and 5.10, it follows that \mathcal{NISZK} is closed under \mathcal{AC}^0 truth-table reductions. Applying Proposition 5.4 ($\mathsf{ED} \leq_{\mathcal{AC}^0-\mathsf{tt}} \mathsf{EA}$) and Lemma 3.1 ($\mathsf{EA} \in \mathcal{NISZK}$), we conclude that $\mathsf{ED} \in \mathcal{NISZK}$. Since ED is complete for \mathcal{SZK} [27] and \mathcal{NISZK} is closed under Karp reductions (Lemma 5.8), we have $\mathcal{SZK} \subset \mathcal{NISZK}$. As $\mathcal{NISZK} \subset \mathcal{SZK}$ is true from the definition of \mathcal{NISZK} , we conclude that $\mathcal{NISZK} = \mathcal{SZK}$.

Finally, we deduce Theorem 1.6, which gives a number of conditions equivalent to $\mathcal{NISZK} = \mathcal{SZK}$.

Proof of Theorem 1.6:

- $1 \Rightarrow 3$. This follows from the result of [39] that \mathcal{SZK} is closed under \mathcal{NC}^1 truthtable reductions.
- $3 \Rightarrow 2 \Rightarrow 1$. The first is trivial and the second is Proposition 5.11.
- $1 \Leftrightarrow 4$. This follows from Theorem 1.3 (which asserts that that EA and SDU are complete for \mathcal{NISZK}), the fact that ED and SD are complete for \mathcal{SZK} [38, 27], and Lemma 5.8 (that \mathcal{NISZK} is closed under Karp reductions).
- $2 \Leftrightarrow 5$. This follows from Theorem 1.3 (that EA and SDU are complete for \mathcal{NISZK}) and Lemma 5.8 (that \mathcal{NISZK} is closed under Karp reductions).

Acknowledgments

The first author was supported by DARPA grant DABT63-96-C-0018. The second and third authors were supported by DOD/NDSEG Graduate Fellowships and partially by DARPA grant DABT63-96-C-0018. We thank the CRYPTO '99 program committee and reviewers for helpful comments on the presentation.

References

- [1] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the Thirty Third Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.
- [2] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs. In *Proceedings* of the Thirty Third Annual Symposium on Foundations of Computer Science, pages 2–13, 1992.
- [3] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 194–211. Springer-Verlag, 1990, 20–24 August 1989.
- [4] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 482–493, 1990.

- [5] Mihir Bellare and Phillip Rogaway. Non-interactive perfect zero-knowledge. Unpublished manuscript, June 1990.
- [6] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. SIAM Journal on Computing, 20(6):1084–1118, December 1991.
- [7] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the Twentieth Annual* ACM Symposium on Theory of Computing, pages 103–112, Chicago, Illinois, 2–4 May 1988.
- [8] Ivan Damgård. Interactive hashing can simplify zero-knowledge protocol design. In Proceedings of Crypto '95, Lecture Notes in Computer Science, volume 403, pages 100-109. Springer-Verlag, 1994.
- [9] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs. dishonest verifier in public coin zero-knowledge proofs. In *Proceedings* of Crypto '95, Lecture Notes in Computer Science, volume 403. Springer-Verlag, 1995.
- [10] Ivan Damgård, Oded Goldreich, and Avi Wigderson. Hashing functions can simplify zero-knowledge protocol design (too). Technical Report RS-94-39, BRICS, November 1994. See Part 1 of [9].
- [11] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-efficient non-interactive zero-knowledge (extended abstract). In Pierpaolo Degano, Robert Gorrieri, and Alberto Marchetti-Spaccamela, editors, Automata, Languages and Programming, 24th International Colloquium, volume 1256 of Lecture Notes in Computer Science, pages 716–726, Bologna, Italy, 7–11 July 1997. Springer-Verlag.
- [12] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image Density is complete for non-interactive-SZK. In Automata, Languages and Programming, 25th International Colloquium, Lecture Notes in Computer Science, pages 784–795, Aalborg, Denmark, 13–17 July 1998. Springer-Verlag. See [13].
- [13] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image Density is complete for non-interactive-SZK, May 1999. Preliminary draft of full version.
- [14] Alfredo De Santis, Giovanni Di Crescenzo, and Guiseppe Persiano. The knowledge complexity of quadratic residuosity languages. Theoretical Computer Science, 132(1-2):291-317, 26 September 1994.
- [15] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, Advances in Cryptology—CRYPTO '87, volume 293 of Lecture Notes in Computer Science, pages 52–72. Springer-Verlag, 1988, 16–20 August 1987.
- [16] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In S. Goldwasser, editor, Advances in Cryptology—CRYPTO '88, volume 403 of Lecture Notes in Computer Science, pages 269–282. Springer-Verlag, 1990, 21–25 August 1988.
- [17] Giovanni Di Crescenzo, Tatsuaki Okamoto, and Moti Yung. Keeping the SZK-verifier honest unconditionally. In Advances in Cryptology—CRYPTO '97, pages 31–45, 1997.
- [18] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing, pages 542–552, New Orleans, Louisiana, 6–8 May 1991.
- [19] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, pages 409–418, 1998.

- [20] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In 31st Annual Symposium on Foundations of Computer Science, volume I, pages 308–317, St. Louis, Missouri, 22–24 October 1990. IEEE.
- [21] Lance Fortnow. The complexity of perfect zero-knowledge. In Silvio Micali, editor, Advances in Computing Research, volume 5, pages 327–343. JAC Press, Inc., 1989.
- [22] Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm. *Journal of Cryptology*, 6:97–116, 1993.
- [23] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the Association for Computing Machinery*, 38(1):691–729, 1991.
- [24] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, Winter 1994.
- [25] Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, pages 399–408, 1998.
- [26] Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero knowledge be made non-interactive? or On the relationship of SZK and NISZK. Submitted to ECCC, May 1999. http://www.eccc.uni-trier.de/eccc/.
- [27] Oded Goldreich and Salil Vadhan. Comparing entropies in statistical zeroknowledge with applications to the structure of SZK. In Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity, pages 54–73, Atlanta, GA, May 1999. IEEE Computer Society Press.
- [28] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. Journal of Computer and System Sciences, 28(2):270–299, 1984.
- [29] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal on Computing, 18(1):186–208, February 1989.
- [30] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, Washington, 15–17 May 1989.
- [31] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, Winter 1998.
- [32] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–123, December 1975.
- [33] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proofs. In *Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science*, pages 1–10, 1990.
- [34] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 427–437, Baltimore, Maryland, 14–16 May 1990.
- [35] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. In *Proceedings of the Twenty Eighth Annual ACM Symposium on the Theory of Computing*, 1996. See also preprint of full version, Oct. 1997.
- [36] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Thirty Second Annual Symposium on Foundations of Computer Science*, pages 133–138, 1991.
- [37] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proceedings of the Second Israel Symposium on Theory of Computing and Systems*, 1993.

- [38] Amit Sahai and Salil Vadhan. A complete promise problem for statistical zero-knowledge. In *Proceedings of the Thirty Eighth Annual Symposium on Foundations of Computer Science*, pages 448–457, 1997.
- [39] Amit Sahai and Salil Vadhan. Manipulating statistical difference. In Panos Pardalos, Sanguthevar Rajasekaran, and José Rolim, editors, Randomization Methods in Algorithm Design (DIMACS Workshop, December 1997), volume 43 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 251–270. American Mathematical Society, 1999.
- [40] Adi Shamir. IP=PSPACE. In Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science, pages 11–15, 1990.
- [41] Andrew C. Yao. Theory and application of trapdoor functions. In Proceedings of the Twenty Third Annual Symposium on Foundations of Computer Science, pages 80–91, 1982.

A Definitions

Following [22], we extend the standard definition of interactive proof systems to promise problems $\Pi = (\Pi_{YES}, \Pi_{NO})$. That is, we require the completeness condition to hold for YES instances (i.e., $x \in \Pi_{YES}$), require the soundness condition to hold for NO instances (i.e., $x \in \Pi_{NO}$), and do not require anything for inputs which violate the promise (i.e., $x \notin \Pi_{YES} \cup \Pi_{NO}$).

We are mainly interested in such proof systems which are statistical zero knowledge:

Definition A.1. (Statistical Zero Knowledge – \mathcal{SZK}): Let (P, V) be an interactive proof system for a promise problem $\Pi = (\Pi_{YES}, \Pi_{NO})$.

- We denote by $\langle P, V \rangle(x)$ the view of the verifier V while interacting with P on common input x; this consists of the common input, V's internal coin tosses, and all messages it has received.
- (P,V) is said to be (general) statistical zero knowledge if, for every probabilistic polynomial-time V^* , there exists a probabilistic polynomial-time machine (called a simulator), S, and a negligible function $\mu: \mathbb{N} \mapsto [0,1]$ (called the simulator deviation) so that for every $x \in \Pi_{YES}$ the statistical difference between S(x) and $\langle P, V^* \rangle(x)$ is at most $\mu(|x|)$.
- SZK denotes the class of promise problems having statistical zero-knowledge interactive proof systems.

Honest-verifier statistical zero-knowledge proof systems are such where the zero-knowledge requirement is only required to hold for the prescribed/honest verifier V, rather than for every polynomial-time computable V^* . Every honest-verifier statistical zero-knowledge proof system can be transformed into a general statistical zero-knowledge proof system (actually meeting an even stronger zero-knowledge requirement) [25].

On Concurrent Zero-Knowledge with Pre-Processing*

(Extended abstract)

GIOVANNI DI CRESCENZO¹

RAFAIL OSTROVSKY²

 University of California San Diego, and Telcordia Technologies, Inc.
 E-mail: giovanni@cs.ucsd.edu, giovanni@research.telcordia.com
 Telcordia Technologies, Inc.
 445 South Street, Morristown, NJ, 07960-6438, USA.
 E-mail: rafail@research.telcordia.com
 *Copyright © 1999, Telcordia Technologies, Inc. All Rights Reserved.

Abstract. Concurrent Zero-Knowledge protocols remain zero-knowledge even when many sessions of them are executed together. These protocols have applications in a distributed setting, where many executions of the same protocol must take place at the same time by many parties, such as the Internet. In this paper, we are concerned with the number of rounds of interaction needed for such protocols and their efficiency. Here, we show an efficient constant-round concurrent zero-knowledge protocol with preprocessing for all languages in NP, where both the preprocessing phase and the proof phase each require 3 rounds of interaction. We make no timing assumptions or assumptions on the knowledge of the number of parties in the system. Moreover, we allow arbitrary interleavings in both the preprocessing and in the proof phase. Our techniques apply to both zero-knowledge proof systems and zero-knowledge arguments and we show how to extend our technique so that polynomial number of zero-knowledge proofs/arguments can be executed after the preprocessing phase is done.

1 Introduction

The notion of zero-knowledge proof systems was introduced in the seminal paper of Goldwasser, Micali and Rackoff [18]. Since their introduction, zero-knowledge proofs have proven to be very useful as a building block in the construction of cryptographic protocols, especially after Goldreich, Micali and Wigderson [17] have shown that all languages in NP admit zero-knowledge proofs. Due to their importance, the efficiency of zero-knowledge protocols has received considerable attention. One of the aspects of efficiency is the number of rounds of interaction necessary for zero-knowledge proofs, and it was shown that there exist computational zero-knowledge proofs with a constant number of rounds for all languages in NP by Goldreich and Kahan [16].

Previous work. Recently, a lot of attention has been paid to the setting where many concurrent executions of the same protocol take place (say, on the Internet). For example, in the context of identification schemes, this was discussed by Beth and Desmedt [3]. In the context of zero-knowledge, Di Crescenzo constructed proof systems that remain zero-knowledge in a synchronous setting

without timing or complexity assumptions [8]. The general case of concurrent zero-knowledge in an asynchronous setting was considered by Dwork, Naor and Sahai [11]. They showed that assuming that there are some bounds on the relative speed of the processors, one can construct four-round zero-knowledge argument for any language in NP (an argument differs from a proof system in the fact that the soundness property holds only with respect to polynomial-time provers [4]). Dwork and Sahai improved this result by reducing the use of timing assumptions to a preprocessing phase [12]. (Preprocessing was first used in the context of zero-knowledge proofs by De Santis, Micali and Persiano [7].) Recently, Richardson and Kilian [24] presented concurrent zero-knowledge protocols for all languages in NP that do not use timing assumption but require more then constant number of rounds of interaction. More specifically, given a security parameter k, Richardson and Kilian's protocol requires O(k) rounds and allows poly(k) concurrent executions. A negative result was given by Kilian, Petrank and Rackoff [20], who showed that if there exists a 4-round protocol that is concurrent (black-box simulation) zero-knowledge for some language Lthen L is in BPP.

Our results. We consider a distributed model with a preprocessing phase and a proof phase. In this setting, we show how, based on complexity assumptions, and after a three-round pre-processing stage, three-round concurrent zero-knowledge proofs (and arguments) can be constructed for all languages in NP. Our protocol does not require any timing assumptions nor knowledge of the number of parties in the system nor knowledge of the total number of concurrent executions. In the case of proof systems, our protocol is based on the existence of perfectly secure commitment schemes, it is computational zero-knowledge and applies to all public-coin zero-knowledge proof systems. In the case of arguments, our protocol is based on the intractability of computing discrete logarithms modulo primes and it is perfect zero-knowledge. The requirement that we make (which is different than in the work of Dwork and Sahai [12]) is that all the concurrent executions of the pre-processing subprotocols finish before the concurrent executions of the proof protocols begin, although we allow arbitrary interleaving both in the pre-processing and in the proof phase. A different interpretation of our result is that we do not make any timing assumptions, but require a single "synchronization barrier" between the pre-processing stage and the main proof stage of the protocol, where all parties finish the pre-processing stage before the main stage begins. We believe that this setting may be of interest in several applications, since the pre-processing stage does not need the knowledge of the theorem(s) to be proved in the main phase of the protocol.

Tools used. An important tool used in the construction of our schemes is that of equivocable commitment schemes, which we consider in two variants: computationally and perfectly equivocable. The computational variant was first discussed by Beaver in [1], and a first construction was given by Di Crescenzo, Ishai and Ostrovsky in [9] in the common random string model (using a scheme by Naor [21]). Here we present a computationally equivocable and a perfectly equivocable commitment schemes in the interactive model, based on bit commitment schemes by Naor [21] and Pedersen [23], respectively. These schemes might be of independent interest and may have further applications, such as for identification schemes. We remark that a somewhat weaker version of our results could be alternatively derived by considering in our model appropriate modifications of techniques based on coin flipping and non-interactive zero-knowledge proofs, as

those used in [24,12,13], resulting with even smaller round-complexity. However, our techniques apply to perfect zero-knowledge arguments and offer additional efficiency properties since they do not use general NP reductions. Another tool we use is that of straight-line simulation, as formally defined by Dwork and Sahai [12] and also used by Feige and Shamir [14].

2 Notations and Definitions

In this section we give basic notations, we recall the notions of interactive proof systems, zero-knowledge proof systems in the two-party model and formally define concurrent zero-knowledge proof systems with preprocessing.

Probabilistic setting. The notation $x \leftarrow S$ denotes the random process of selecting element x uniformly from set S. Similarly, the notation $y \leftarrow A(x)$, where A is an algorithm, denotes the random process of obtaining y when running algorithm A on input x, where the probability space is given by the random coins (if any) of algorithm A. By $\{R_1; \ldots; R_n : v\}$ we denote the set of values v that a random variable can assume, due to the distribution determined by the sequence of random processes R_1, \ldots, R_n . By $\text{Prob}[R_1; \ldots; R_n : E]$ we denote the probability of event E, after the ordered execution of random processes R_1, \ldots, R_n .

Interactive protocols. Following [18], an interactive Turing machine is a Turing machine with a public input tape, a public communication tape, a private random tape and a private work tape. An interactive protocol is a pair of interactive Turing machines sharing their public input tape and communication tape. The transcript of an execution of an interactive protocol (A,B) is a sequence containing the random tape of B and all messages appearing on the communication tape of A and B. The notation $(y_1, y_2) \leftarrow (A(x_1), B(x_2))(x)$ denotes the random process of running interactive protocol (A,B), where A has private input x_1 , B has private input x_2 , x is A and B's common input, y_1 is A's output and y_2 is B's output, where any of x_1, x_2, y_1, y_2, x can be an empty string; the notation $y \leftarrow (A(x_1), B(x_2))(x)$ is an abbreviation for $(y, y) \leftarrow (A(x_1), B(x_2))(x)$. We assume wlog that the output of both parties A and B at the end of an execution of protocol (A,B) contains a transcript of the communication exchanged between A and B during such execution. An interactive protocol with preprocessing is a pair of interactive protocols ((A1,B1),(A2,B2)). The mechanics of an interactive protocol with preprocessing is divided in two phases, as follows. In a first phase, called the preprocessing phase, the first pair (A1,B1) is executed; at the end of this phase a string α is output by A1 and given as private input to A2, and a string β is output by B1 and given as private input to B2. Now, an input string x is given as common input to A2 and B2, and the second pair (A2,B2)is executed. In this paper we will be concerned with two types of interactive protocols: proof systems and arguments, which we now describe.

Interactive proof systems and arguments. An interactive proof system for a language L is an interactive protocol in which, on input a string x, a computationally unbounded prover convinces a polynomial-time bounded verifier that x belongs to L. The requirements are two: completeness and soundness. Informally, completeness states that for any input $x \in L$, the prover convinces the verifier with very high probability. Soundness states that for any $x \notin L$ and any prover, the verifier is convinced with very small probability. A formal

definition can be found in [18,15]. An argument is defined similarly to a proof system, the only difference being that the prover is assumed to be polynomially bounded (see [4]).

Zero-knowledge proof systems in the two-party model. A zero-knowledge proof system for a language L is an interactive proof system for L in which, for any $x \in L$, and any possibly malicious probabilistic polynomial-time verifier V', no information is revealed to V' that he could not compute alone before running the protocol. This is formalized by requiring, for each V', the existence of an efficient simulator $S_{V'}$ which outputs a transcript 'indistinguishable' from the view of V' in the protocol. There exist three notions of zero-knowledge, according to the level of indistinguishability: computational, statistical and perfect. We refer the reader to [18,15] for the definitions of computational, statistical and perfect indistinguishability between distributions and for the definitions of computational, statistical and perfect zero-knowledge proof systems.

The concurrent model for zero-knowledge proof systems. Informally speaking, the concurrent model describes a distributed model in which several parties can run concurrent executions of some protocols. In real-life distributed system, the communication is not necessarily synchronized; more generally, the model makes the worst-case assumption that the communication in the system happens in an asynchronous manner; this means that there is no fixed bound on the amount of time that a message takes to arrive to its destination. This implies that, for instances, the order in which messages are received during the execution of many concurrent protocols can be arbitrarily different from the order in which they were sent. Such variation in the communication model poses a crucial complication into designing a zero-knowledge protocol, since the possibly arbitrary interleaving of messages can help some adversary to break the zero-knowledge property. In fact, as a worst case assumption, one may assume that the ordering in which messages are received can be decided by the adversary. Now we proceed more formally.

We consider a distributed model, with two distinguished sets of parties: a set $\mathcal{P} = \{P_1, \dots, P_q\}$ of provers and a set $\mathcal{V} = \{V_1, \dots, V_q\}$ of verifiers, where P_i is connected to V_i , for $i=1,\ldots,q$. Let (A,B) be a zero-knowledge proof system. At any time a verifier V_i may decide to run protocol (A,B); therefore, for any fixed time, there may be several pairs of prover and verifier running (A,B). The adversary \mathcal{A} is allowed to corrupt all the verifiers. Then \mathcal{A} can be formally described as a probabilistic polynomial time algorithm that, given the history so far, returns an index i and a message m_i , with the meaning that the (corrupted) verifier V_i , for $i \in \{1, \ldots, q\}$, is sending message m_i to prover P_i . We assume wlog that P_i is required to send his next message after receiving m_i and before receiving a new message from A. We now define the view of the adversary A. First we define a qconcurrent execution of (A,B) as the possibly concurrent execution of q instances of protocol (A,B), where all verifiers are controlled by A. Also, we define the q-concurrent transcript of a q-concurrent execution of (A,B) as a sequence containing the random tapes of verifiers V_1, \ldots, V_q and all messages appearing on the communication tapes of P_1, \ldots, P_q and V_1, \ldots, V_q , where the ordering of such messages is determined by the adversary corrupting all the verifiers. The notation $(T; y_{11}, y_{12}, \dots, y_{1q}, y_{2q}) \leftarrow ((P_1(p_1), V_1(v_1))(x_1), \dots, (P_q(p_q), V_q(v_q))(x_q))$ denotes the random process of running a q-concurrent execution of interactive protocol (A,B), where each P_i has private input x_i , each V_i has private input

 v_i , x_i is P_i and V_i 's common input, y_{1i} is P_i 's output, y_{2i} is V_i 's output, and T is the q-concurrent transcript of such execution (we assume wlog that the output of both parties P_i and V_i at the end of an execution of the interactive protocol (A,B) contains a transcript of the communication exchanged between P_i and V_i during such execution). Then the view of A, denoted as $View_A(x)$, is the transcript T output by the random process $(T;y_{11},y_{12},\ldots,y_{1q},y_{2q}) \leftarrow ((P_1(p_1),A(v_1))(x_1),\ldots,(P_q(p_q),A(v_q))(x_q))$, where $x=(x_1,\ldots,x_q)$. Finally, a proof system (A,B) for language L is concurrent zero-knowledge if for any probabilistic polynomial time algorithm A, there exists an efficient algorithm S_A such that for any polynomial $q(\cdot)$, and any x_1,\ldots,x_q , where q=q(n), and $|x_1|=\cdots=|x_q|=n$, the two distributions $S_A(x)$ and $View_A(x)$ are indistinguishable.

Concurrent zero-knowledge proof systems with preprocessing. We now consider a variant of the above distributed model, in which we would like to consider concurrent zero-knowledge proof systems with preprocessing. In this variant a concurrent execution of an interactive protocol with preprocessing ((A1,B1),(A2,B2)) is divided into two phases: a preprocessing phase and a proof phase. In the preprocessing phase there is a concurrent execution (in the sense defined before) of the preprocessing pair (A1,B1) and in the proof phase there is a concurrent execution of the proof pair (A2,B2). The requirements are, as before, completeness, soundness and concurrent zero-knowledge. Now we give a formal definition of concurrent zero-knowledge proof systems with preprocessing.

Definition 1. Let (A,B)=((A1,B1),(A2,B2)) be an interactive protocol with preprocessing. We say that (A,B) is a concurrent (computational) (statistical) (perfect) zero-knowledge proof system with preprocessing for language L if the following holds:

- 1. Completeness: For any $x \in L$, it holds that $\text{Prob}[(\alpha, \beta) \leftarrow (A1, B1)(1^{|x|}); (t, (t, out)) \leftarrow (A2(\alpha), B2(\beta))(x) : out = \text{ACCEPT}] \ge 1 2^{-|x|}.$
- 2. Soundness: For any $x \notin L$, and any (A1',A2'), it holds that $\text{Prob}[(\alpha,\beta) \leftarrow (A1',B1)(1^{|x|});(t,(t,out)) \leftarrow (A2'(\alpha),B2(\beta))(x): out = \text{ACCEPT}] < 2^{-|x|}$.
- 3. Concurrent Zero-Knowledge: For each probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists an expected polynomial time simulator algorithm $S_{\mathcal{A}}$ such that for any polynomial $q(\cdot)$, for each $x_1, \ldots, x_q \in L$, where $|x_1| = \cdots = |x_q| = n$ and q = q(n), the two distributions $S_{\mathcal{A}}(x)$ and $View_{\mathcal{A}}(x)$ are (computationally) (statistically) (perfectly) indistinguishable, where

$$View_{\mathcal{A}}(\boldsymbol{x}) = \{ (T_1, \alpha_1, \beta_1, \dots, \alpha_q, \beta_q) \leftarrow ((P_1, \mathcal{A}_1), \dots, (P_q, \mathcal{A}_1))(1^n);$$

$$(T_2, \cdot, \cdot, \dots, \cdot, \cdot) \leftarrow ((P_1(\alpha_1), \mathcal{A}_2(\beta_1))(x_1), \dots, (P_q(\alpha_q), \mathcal{A}_2(\beta_q))(x_q)) :$$

$$(T_1, T_2, \beta_1, \dots, \beta_q) \}$$

denotes the view of A on input $x = (x_1, ..., x_q)$, and where $P_1, ..., P_q$ run algorithms A1 in the preprocessing phase and A2 in the proof phase.

3 Cryptographic tools and a class of languages

We briefly review some cryptographic tools as commitment schemes and coin flipping protocols. Then we discuss a class of languages having a certain type of zero-knowledge protocols, which will be used in this paper.

Commitment schemes. Informally speaking, a commitment scheme (Alice, Bob) is a two-phase interactive protocol between two probabilistic polynomial time parties Alice and Bob, called the sender and the receiver, respectively, such that the following is true. Alice commits to his bit b in the first phase (called the commitment phase); in the second phase (called the decommitment phase) Alice convinces Bob of the value of the bit b Alice had committed to in the first phase (if Bob is not convinced, he outputs a special string \perp). A commitment scheme has three requirements. First, if Alice and Bob behave honestly, then at the end of the decommitment phase Bob is convinced that Alice had committed to bit b with high probability (this is the correctness requirement). Then, no matter which polynomial-time computable strategy Bob uses in the commitment phase, Bob is not able to guess such bit with probability significantly better than its a priori probability at the end of such phase (this is the *security* property). Finally, for any strategy played by Alice, the probability that he can later decommit both as 0 and as 1 is negligible (this is the binding property). There are two main variants of commitment schemes in the literature, with respect to the type of security guaranteed: computationally secure (i.e., the security property holds against a polynomial time bounded receiver, while the binding property holds against an unrestricted sender) and perfectly secure (i.e., security holds against an unrestricted receiver, while binding holds against polynomial time bounded sender). In this paper we will consider both variants.

Coin flipping protocols. A coin flipping protocol [5] is a protocol among two parties, Alice and Bob, who exchange messages. At the end of the protocol, both parties have a bit that is uniformly distributed, no matter how each of them tries to influence its distribution. Standard constructions for coin flipping protocols are obtained by using bit commitment schemes, as follows. Alice commits to a random bit a, Bob sends a random bit b to Alice, then Alice reveals her bit a; the result of the coin flipping protocol is $c = a \oplus b$.

Three-round public-coin honest-verifier zero-knowledge protocols. We will consider a class of languages having a special type of zero-knowledge protocol; namely, a protocol having the following properties: 1) the protocol has three rounds; namely, the prover sends a message (called *first message*) to the verifier, the verifier sends a message (called *challenge*) to the prover, the prover sends a final message (called *answer*) to the verifier who decides whether to accept or not; 2) it is public-coin; namely, the challenge consists of some random bits sent by the verifier; 3) it is honest-verifier zero-knowledge; namely, the zero-knowledge requirement holds only with respect to a verifier which follows its program. In the literature there are several examples of protocols satisfying the above properties, and the class of languages having such protocols is quite large. For instance, all languages in NP have a computational zero-knowledge proof system (e.g., [17, 5]) with these properties, and all random self-reducible languages and formula compositions over them have perfect zero-knowledge proof systems with these properties ([18, 17, 25, 6]).

4 Equivocable commitment schemes

We recall the notion of equivocable commitment schemes, by presenting two variants of them: computationally and perfectly equivocable commitment schemes. Then we present an example of a computationally equivocable commitment

scheme under the assumption of the existence of any one-way permutation and an example of a perfectly equivocable commitment scheme under the assumption of intractability of computing discrete logarithms.

Definition of equivocable commitment schemes. Informally speaking, a bit commitment scheme is equivocable if it satisfies the following additional requirement. There exists an efficient algorithm, called the simulator, which outputs a transcript leading to a 'fake' commitment such that: (a) the 'fake' commitment can be decommitted both as 0 and as 1, and (b) the simulated transcript is indistinguishable from a real execution of the protocol. The extension to equivocable string commitment schemes is straightforward. The two types of equivocability, computational and perfect, differ according to the type of indistinguishability in (b). A formal definition follows.

Definition 2. Let (A,B) be a commitment scheme. We say that (A,B) is computationally equivocable (resp., perfectly equivocable) if for any probabilistic polynomial time algorithm B', there exists a probabilistic polynomial time algorithm A' such that:

1. Equivocability. For all constants c, all sufficiently large n, any string $s \in \{0,1\}^k$, where k = k(n) for some polynomial $k(\cdot)$, it holds that $|p_0 - p_1| \le n^{-c}$, where p_0, p_1 are, respectively,

$$\text{Prob}[(\alpha,\beta) \leftarrow M_{B'}(1^n,1^k); (t,(t,v)) \leftarrow M_{B'(\beta)}(\alpha,s,1^n) : v = s],$$

$$\text{Prob}[(\alpha,\beta) \leftarrow (A(s),B')(1^n,1^k); (t,(t,v)) \leftarrow (A(\alpha,s),B'(\beta))(1^n,1^k) : v = s].$$
 (resp., it holds that $p_0 = p_1$.)

2. Indistinguishability. For any string $s \in \{0,1\}^k$, where k = k(n) for some polynomial $k(\cdot)$, the distributions T(M) and T(A) are computationally (resp., perfectly) indistinguishable, where

$$T(M) = \{(\alpha, \beta) \leftarrow M_{B'}(1^n, 1^k); (t, (t, s)) \leftarrow M_{B'(\beta)}(\alpha, s, 1^n) : (\beta, (t, s))\},$$

$$T(A) = \{(\alpha, \beta) \leftarrow (A(s), B')(1^n, 1^k); (t, (t, s)) \leftarrow (A(\alpha, s), B'(\beta))(1^n, 1^k) : (\beta, (t, s))\}.$$

Equivocable bit-commitment schemes have properties similar to chameleon or trapdoor commitment schemes (see, e.g., [4]); a main difference is that in equivocable commitment schemes one among the two requirements of binding and security holds with respect to a computationally unlimited adversary (as opposed to only polynomial-time bounded). Computationally equivocable bit-commitment schemes have been first discussed in [1], who observed the somewhat paradoxical requirement that such schemes need to satisfy. In [9] it was shown that the implementation in the common random string model of the bit commitment scheme from [21] is computationally equivocable. Here we show in the interactive model that a variation of the scheme in [21] is a computationally equivocable commitment scheme and a variation of the scheme in [23] is a perfectly equivocable commitment scheme.

A computationally equivocable commitment scheme. We will present a scheme (A,B) by combining the scheme in [21] with a coin flipping protocol; we describe the scheme for a single bit, since the extension to a many-bit string is straightforward. The bit commitment scheme in [21] consists in a 2-round

commitment phase and a 1-round decommitment phase. In the commitment phase, the receiver sends a 3n-bit random string R to the committer and the committer replies with some pseudo-random message com, where n is a security parameter. In the decommitment phase, the committer sends a decommitment message dec which allows, together with R and com, the receiver to be convinced that the committed bit was b. The variation we consider here consists in the following. First of all committer and receiver run a coin flipping protocol, whose output we denote by r. Then they continue the protocol as in [21], but using the output r of the coin flipping protocol as string R is used in the original protocol. A more formal description follows. We will denote by n a security parameter, and assume that the parties share a perfectly secure commitment scheme (C,D) and a pseudo-random generator G.

The Commitment Protocol (A,B)

Input to A: a bit b.

Commitment Phase:

- 1. B uniformly chooses string $u \in \{0, 1\}^{3n}$; B commits to it using scheme (C,D) (possibly interacting with A);
- 2. A uniformly chooses string $v \in \{0, 1\}^{3n}$ and sends it to B;
- 3. B decommits u to A using scheme (C,D) (possibly interacting with A);
- 4. If u is not properly decommitted, A halts.

A uniformly chooses $s \in \{0,1\}^n$ and computes z = G(s);

A sets com = z if b = 0 or $com = z \oplus u \oplus v$ if b = 1, and sends com to B.

Decommitment Phase:

- 1. A sends s to B;
- 2. B outputs: 0 if G(s) = com, 1 if $G(s) = com \oplus u \oplus v$, \bot otherwise.

We obtain the following

Theorem 1. Assuming the existence of pseudo-random generators and perfectly-secure commitment schemes, the protocol (A,B) is a computationally equivocable and computationally secure commitment scheme.

Proof. The correctness and the security property of (A,B) directly follow from those of the scheme in [21]. Now we consider the binding property. Note that since B commits to u using a perfectly-secure commitment scheme, no infinitely powerful A' can guess u better than guessing at random. Therefore, for any A', the distribution of $u \oplus v$ is uniform over $\{0,1\}^{3n}$, and we can directly apply the analysis of [21] to conclude that (A,B) satisfies the binding property. Now we show that (A,B) is computationally equivocable. Informally, we present an efficient simulator M, which can run the commitment phase in such a way that it can later decommit both as a 0 and as a 1.

The algorithm M. On input 1^n , M uniformly chooses $u \in \{0,1\}^{3n}$ and two seeds $s_0, s_1 \in \{0,1\}^n$, and computes $z_0 = G(s_0)$ and $z_1 = G(s_1)$. Then it runs the commitment phase as follows: it receives a commitment to v from B', it sends u to B' and it receives the decommitment of v from B'; at this point, M either halts (if decommitments were not appropriate) or rewinds B' to the state right after B' committed to v, sets $u' = z_0 \oplus z_1 \oplus v$ and sends u' to B'; now M again receives the decommitment of v from B'; at this point, M either halts (if decommitments were not appropriate) or continues the simulation, and the result of the coin flipping protocol will be $u' \oplus v$; finally, in order to commit, M sends $com = z_0$ to B; in order to decommit as v0, v1, v2, v3, v3, v4, v5, v5, v6, v8, v8, v8, v9, v9,

M's output is computationally indistinguishable from a real execution of (A,B). Note that in M's output the result of the coin flipping protocol is equal to $G(s_0) \oplus G(s_1)$ while in an execution of (A,B) such output is random. In particular, the same holds for the string sent to the receiver during the coin flipping protocol. This can be used to show, exactly as already done in [9], that if there exists an efficient algorithm distinguishing M's output from a real execution of (A,B), then there exists an efficient algorithm that distinguishes uniformly distributed strings from outputs of G. This contradicts the assumption that G is a pseudorandom generator.

M can decommit both as θ and as 1. During the execution of M there are two executions of the coin flipping protocol, one before and one after the rewinding of B'; we will call them the first and the second execution of the coin flipping protocol, respectly; we note that M can halt both during the first and during the second execution of the coin flipping protocol because of an inappropriate decommitment by B'. Instead, if the result of the second execution of the coin flipping protocol between M and B' is $u' \oplus v = z_0 \oplus z_1$, then s_0 and s_1 are valid decommitment keys as 0 and 1, respectively, for the commitment $com = z_0$. Therefore, the probability p_0 , as defined in item 1 of Definition 2 is equal to the probability that the output of the second execution of the coin flipping protocol is equal to $z_0 \oplus z_1$. Let us define probability q_1 as the probability that M does not halt in the first execution of the coin flipping protocol, and probability q_0 as the probability that M halts neither in the first nor in the second execution of the coin flipping protocol because of some inappropriate decommitment of B'. We now observe three facts.

The first fact is that $p_1 = q_1$. This follows by observing the probabilistic experiments in the definition of p_1 and q_1 are the same; specifically, notice that the string u is uniformly chosen and B' decommits properly in both experiments.

The second fact we observe is that $p_0 = q_0$. This follows by observing the probabilistic experiments in the definition of p_0 and q_0 are the same; specifically, observe that M is successful in creating a fake commitment if and only if M does not halt neither in the first execution nor in the second execution of the coin flipping protocol.

The third fact is that $q_1 - q_0$ is negligible. Assume not. Then B' could be used to efficiently distinguish between the random string u sent by M during the first execution of the coin flipping protocol and the string u' sent by M during the second execution (since B' correctly decommits in the first case and incorrectly in the second). This implies that B' can be, in turn, used to contradict the fact that G is a pseudo-random generator (exactly as in the proof that M's output is computationally indistinguishable from a real execution of (A,B)), thus giving a contradiction.

By combining the above three facts, we have that $|p_1 - p_0| \le |q_1 - q_0|$, and is therefore negligible.

We remark that the perfectly-secure commitment scheme (C,D) used in the above construction can be implemented assuming the existence of any one-way permutation using the scheme in [22]. Therefore, the weakest assumption under which the described scheme (A,B) is computationally equivocable is the existence of one-way permutations.

A perfectly equivocable commitment scheme. Our perfectly equivocable scheme is based on a perfectly-secure commitment scheme in [23]; this scheme

uses some elementary number theoretic definition concerning discrete logarithms, which we briefly review.

Discrete logarithms. Let p,q be primes such that p=2q+1 and let G_q denote the only subgroup of Z_p^* of order q. We note that it can be efficiently decided whether an integer a is in Z_q , by checking that $a^q \equiv 1 \mod p$. Moreover, any element of G_q different from 1 generates such subgroup. For any $a,b \in G_q$, if $b \neq 1$ the discrete logarithm of a in base b is the integer x such that $b^x = a \mod p$.

The commitment scheme in [23]. Given primes p,q such that p=2q+1 and $g,h\in G_q$, the commitment scheme goes as follows. In order to commit to an integer $s\in Z_q$, the committer uniformly chooses $r\in Z_q$, computes $com=g^sh^r \bmod p$, and sends com to the receiver. In order to decommit com as m, the committer sends s,r to the receiver, who checks that $com=g^sh^r \bmod p$. The perfect security property of this scheme follows from the fact that com is uniformly distributed in G_q ; the computational binding property follows from the fact that if a committer is able to successfully decommit a string com both as s and s', then he can compute the discrete logarithm of h in base g.

Our variation. We consider a protocol (A,B) that is a variation of the above scheme, consisting in running first a coin flipping protocol to choose $g, h \in G_q$. A more formal description follows. We will assume that the parties share a computationally secure commitment schemes (C,D).

The Commitment Protocol (A,B)

Input to A: an integer $s \in \mathbb{Z}_q$.

Commitment Phase:

- 1. A uniformly chooses k-bit primes p,q such that p=2q+1 and sends them to B;
- 2. B checks that p = 2q+1 and that p, q are primes and uniformly chooses $u_1, u_2 \in G_q$; B commits to u_1, u_2 using scheme (C,D) (possibly interacting with A);
- 3. A uniformly chooses $v_1, v_2 \in G_q$ and sends them to B;
- 4. B decommits u_1, u_2 to A using scheme (C,D) (possibly interacting with A);
- 5. If u_1, u_2 are not properly decommitted, A halts;
- 6. A and B set $g = u_1v_1 \mod p$ and $h = u_2v_2 \mod p$;
- 7. A uniformly chooses $r \in \mathbb{Z}_q$, computes $com = g^s h^r \mod p$ and sends com to B.

Decommitment Phase:

- 1. A sends s, r to B;
- 2. B outputs: s if $com = g^s h^r \mod p$, \perp otherwise.

We obtain the following

Theorem 2. Assuming the existence of computationally-secure commitment schemes and the intractability of computing discrete logarithms modulo integers of the form p = 2q + 1, for p, q primes, (A,B) is a perfectly equivocable and perfectly secure commitment scheme.

Proof. The correctness and perfect security properties directly follow from the analogue properties of the scheme in [23]. Now we consider the binding property. Note that since B commits to u_1, u_2 using a computationally secure commitment scheme, for any polynomial time A', the distribution of $u_1v_1 \mod p$ and $u_2v_2 \mod p$ has only negligible distance from the uniform distribution. Therefore we can apply the analysis of [23] to conclude that (A,B) is computationally binding assuming that computing discrete logarithms modulo integers of the form

p=2q+1, for q prime, is intractable. The proof that (A,B) is perfectly equivocable can be obtained similarly as for the previous scheme. Similarly, we can construct a simulator M who will run the protocol as A does, learn the string u committed by B, rewind B and force the output of the coin flipping protocol to be a predetermined string z. Such string will be predetermined by M as the string generating $g, h \in G_q$ such that $g^x = h \mod p$, for some $x \in G_q$ chosen by M. Later, a commitment com computed as $com = g^m h^r \mod p$ by M, can be decommitted as $s \in G_q$ in the following way: M randomly chooses $r' \in G_q$ and sets $s = m + x(r - r') \mod p$. It can be seen that (s, r') is a valid decommitment of com as s. The proof continues along the same lines as the proof of Theorem 1. We can define the same probabilities q_0, q_1 and similarly show that $p_0 = q_0$, $q_1 = q_2$ and $p_1 = q_1$, from which we obtain that $p_1 = p_0$. We remark that the assumption of intractability of computing discrete logarithms implies the existence of computationally secure commitment schemes, therefore Theorem 2 holds under the only assumption of the intractability of computing discrete logarithms.

5 Concurrent zero-knowledge proofs with preprocessing

In this section we consider the problem of constructing concurrent zero-knowledge proofs with preprocessing. We present a transformation which applies to all languages having a public-coin honest-verifier zero-knowledge proof systems, including, in particular, all languages in NP. The transformation returns a concurrent zero-knowledge proof system with preprocessing, requiring therefore no timing assumptions nor requiring the parties to know the number of users in the system. Important properties of the resulting proof system include the fact that it has round, communication and computation complexity comparable to those of the starting proof system; specifically, our transformation does not require general NP reductions, and requires a 3-round preprocessing phase and a proof phase with a number of rounds equal to the number of rounds of the original proof system. For simplicity of description, we now present our result for all languages having a 3-round public-coin honest-verifier zero-knowledge proof system. Formally, we achieve the following

Theorem 3. Let L be a language having a 3-round public-coin honest-verifier zero-knowledge proof system. Assuming the existence of pseudo-random generators and perfectly secure commitment schemes, there exists (constructively) a concurrent computational zero-knowledge proof system with preprocessing for L where both preprocessing and proof phases require 3 rounds of communication.

The result underlying our transformation is actually stronger than what stated in the above theorem, providing, for example, additional efficiency properties; various remarks and extensions are discussed in Section 5.2. In the following, we start with an informal description of our transformation, then present a formal description of our concurrent zero-knowledge proof system and then prove its properties.

An informal description. We start by informally describing the main ideas behind our technique. First of all, we should observe that proving a protocol to be concurrent zero-knowledge becomes a problem in the presence of potentially bad interleavings among the polynomially many concurrent executions. In

particular, such interleavings may ask the simulator for too many (eventually, more than polynomial) rewindings in order to be able to succeed in simulating the protocol. Therefore, we use a technique that allows to simulate part of our protocol without performing any rewinding Part of our protocol uses a special zero-knowledge property, similar to a technique used in [12] in a trustedcenter setting and in [14] in the case of arguments. This type of zero-knowledge property, formally defined as straight-line zero-knowledge, does not require the simulator to rewind the adversary. Straight line zero-knowledge arguments have been implemented in [12], using either a trusted center or a preprocessing phase with timing assumptions. In this paper we separate the preprocessing phases of all protocols from the proof phases of all protocols and obtain, without using timing assumptions or trusted centers, that the proof phase of our protocol is straight-line zero-knowledge. This is achieved by using the tool of equivocable commitment. Namely, the prover uses a computationally equivocable commitment scheme to commit to some random string d, receives the challenge c from the verifier, decommits his string d and uses the string $c \oplus d$ as a challenge according to the original protocol (A,B). In this way, during the simulation, he can compute such commitment keys in a way such that he can later open them both as a 0 and as a 1. In particular, he will be able to set string d after he has seen the challenge c from the adversary. Namely, he will be able to set d in such a way that the string $d \oplus c$ matches a challenge consistent with the first message he has already sent and with the output of an execution of the simulator S that he had previously computed. The scheme will use the computationally equivocable commitment presented in Section 4; in particular, the coin flipping subprotocol used in that scheme will be run in the preprocessing phase of this protocol. Notice that the coin flipping protocol is not straight line simulatable, but its concurrent composition can be simulated in time at most quadratic of the number of executions, no matter which interleavings among them are chosen by the adversary.

Formal description. Let L be a language having a 3-round public-coin honest-verifier zero-knowledge proof system (A,B) and let x be the common input, where |x|=n. We denote by S the honest-verifier simulator associated to (A,B), and by (mes,c,ans) a transcript of an execution of (A,B), where mes is the first message, c is the challenge, ans is the answer, and |c|=m. Also, let (C,D) be a perfectly secure commitment scheme and let G be a pseudo-random generator. Now we give a formal description of the preprocessing phase subprotocol (P1,V1) and the proof phase subprotocol (P2,V2) of our concurrent zero-knowledge proof system with preprocessing (P,V).

The Proof System ((P1,V1),(P2,V2))

Input to P1 and V1: 1^n , where n is a positive integer.

Instructions for P1 and V1 (preprocessing phase):

- 1. V1 uniformly chooses $u_1, \ldots, u_m \in \{0, 1\}^{3n}$ and commits to them using scheme (C,D) (possibly interacting with P1);
- 2. P1 uniformly chooses $v_1, \ldots, v_m \in \{0, 1\}^{3n}$ and sends them to V1;
- 3. V1 decommits u_1, \ldots, u_m using scheme (C,D) (possibly interacting with P1);
- 4. If u_1, \ldots, u_m are not properly decommitted, P1 halts;
- 5. P1 sets α equal to the transcript so far and outputs: α ;

6. V1 sets β equal to the transcript so far and outputs: β .

Input to P2: x, α , where |x| = n. Input to V2: x, β , where |x| = n. Instructions for P2 and V2 (proof phase):

- 1. P2 sets mes = A(x) and uniformly chooses $d_1, \ldots, d_m \in \{0, 1\}$;
- 2. For i = 1, ..., m,

```
P2 uniformly chooses s_i \in \{0, 1\}^n and computes z_i = G(s_i);
P2 sets com_i = z_i if d_i = 0 or com_i = z_i \oplus u_i \oplus v_i if d_i = 1;
```

P2 sends $mes, com_1, \ldots, com_m$ to B.

- 3. V2 uniformly chooses $b \in \{0,1\}^m$ and sends b to P_2 ;
- 4. P2 decommits d_1, \ldots, d_m by sending s_1, \ldots, s_m to V_2 ;
- 5. P2 sets $d = d_1 \circ \cdots \circ d_m$, $c = a \oplus d$, ans = A(x, mes, c) and sends ans to V_2 ;
- 6. V2 checks that $com_i = G(s_i)$ if $d_i = 0$ and $com_i = G(s_i) \oplus u_i \oplus v_i$ if $d_i = 1$; if any of these checks fails then V2 halts;
- 7. If B(x, mes, c, ans) = ACCEPT then V2 outputs: ACCEPT else V2 outputs: REJECT.

We remark that the protocol does not require any information about the input to be known at preprocessing phase, other than its length.

5.1 Properties of our protocol

We now prove the properties of the described protocol. Clearly the verifier's algorithms V1 and V2 can be performed in probabilistic polynomial time. The completeness requirement directly follows from the correctness of the equivocable commitment scheme and the completeness of the protocol (A,B). The soundness requirement directly follows from the perfect security of the commitment scheme (C,D) and the analysis in [21]. Now we see that the requirement of concurrent zero-knowledge is satisfied.

Concurrent Zero-Knowledge. In the following we describe an efficient simulator Sim which interacts with a probabilistic polynomial-time adversary \mathcal{A} who corrupts all verifiers V_1, \ldots, V_q . We will show that for all $x_1, \ldots, x_q \in L$, the distributions $\operatorname{Sim}_{\mathcal{A}}(\boldsymbol{x})$ and $\operatorname{View}_{\mathcal{A}}(\boldsymbol{x})$ are computationally indistinguishable, where $\boldsymbol{x} = (x_1, \ldots, x_q)$.

The simulator Sim. We start with an informal description of Sim. The first step of the simulator Sim is to run the preprocessing phase of protocol (P,V), where Sim runs algorithm P1 and \mathcal{A} runs V1. This step is done to check whether \mathcal{A} decommits all the commitments in a proper way. If this is the case, then Sim continues the simulation; otherwise, namely, if there exists at least one commitment that is not open properly from \mathcal{A} , Sim outputs the transcript seen until then and halts. In this step Sim also keeps track of which verifier sent its commitment message first, call it V_1 , and which were its decommitments.

The second step of the simulator Sim consists in performing the simulation of the concurrent executions in the preprocessing phase. The algorithm Sim repeats for each verifier V_i , $i \in \{1, \ldots, q\}$, the following four substeps, given a 'current' verifier V_i , and the value of its decommitted strings. First, it rewinds

the adversary \mathcal{A} until right after V_i had sent its commitments. Then, using its knowledge of the decommitted strings from V_i , he sends some pseudo-random strings which will allow later to send commitments that can be later opened both as 0 and as 1. Now, Sim will continue the simulation of the preprocessing phase by running the (polynomial-time) algorithm P1 and waiting for the next verifier sending its commitment message. Once, he has found such verifier, he will continue the simulation of the preprocessing phase by running P1 and waiting for the decommitments by such verifier, who becomes then the current verifier in the next execution of these four substeps. If at any time, a verifier sends an inappropriate decommitment, Sim just halts.

The third step of the simulator Sim consists in performing the simulation of the concurrent executions in the proof phase. Note that after the second step is over, if Sim has not halted, in all proofs Sim can compute commitments that he can open both as 0 and as 1. Therefore, the proof can now be simulated without rewindings of any verifier, since, after seeing the message from the verifier, Sim can set the challenge any value it likes by just properly decommitting the equivocable commitments.

A formal description is in Figure 5.1. We have the following

Lemma 1. For all probabilistic polynomial time algorithms \mathcal{A} , the algorithm $Sim_{\mathcal{A}}$ runs in expected polynomial time.

Proof. Let us first consider the simulation of the preprocessing phase. From its formal description, we first observe that Sim runs once the preprocessing phase by running algorithm P1; this step can be run in polynomial time since algorithm P1 also runs in polynomial time. Then we observe that Sim rewinds the algorithm \mathcal{A} at most q(n) times, that is, a number of times polynomial in n, since q is a polynomial. During each rewinding Sim clearly runs at most a polynomial number of steps, since G is computable in polynomial time. Finally, it is easy to see that the simulation of the proof phase also takes at most a polynomial number of steps.

Lemma 2. For all probabilistic polynomial time algorithms \mathcal{A} , all polynomials q, and all $x_1, \ldots, x_q \in L$, the distribution $\operatorname{Sim}_{\mathcal{A}}(x)$ is computationally indistinguishable from $\operatorname{View}_{\mathcal{A}}(x)$, where $x = (x_1, \ldots, x_q)$.

Sketch of Proof. We consider three cases according to whether the adversary decommits its commitments in a proper way in various steps of algorithm Sim. The first case we consider is the one in which the adversary \mathcal{A} always decommits its commitments in a proper way. All messages sent by \mathcal{A} are clearly equally distributed in the simulation and in the proof since they are computed in the same way. Now, let us consider the messages from the provers. We observe that the triple (mes_i, c_i, ans_i) is output by the simulator S for (A,B) and therefore its distribution is computationally indistinguishable from the distribution of the same triple in the proof. The only remaining messages to consider are the strings $v_{i,j}$ in the preocessing phase, the commitments $z_{i,j,}$ and the associated decommitments $s_{i,j,}$ in the proof phase. From the description of Sim and P1, we can see that the distributions of such strings are clearly different; for instance, the strings $v_{i,j}$ are random in a real proof and pseudo-random in the simulated execution. However, notice that each triple $(v_{i,j}, z_{i,j,\cdot}, s_{i,j,\cdot})$ is part of the transcript

The Algorithm Sim

Input to Sim: x_1, \ldots, x_q , where $|x_i| = n$.

```
Instructions for Sim (preprocessing phase):
```

1. Run the preprocessing subprotocol (P1,V1), playing as P1 and interacting with \mathcal{A} who plays as V1; (if \mathcal{A} ever decommits inappropriately, output the transcript obtained until then and halt;)

```
call V_1 the first verifier sending its commitments (according to scheme (C,D));
set i=1 and let u_{1,1},\ldots,u_{1,m}\in\{0,1\}^{3n} be the strings decommitted from V_1;
```

2. repeat while i < q:

```
rewind \mathcal{A} until right after V_i has sent its commitments;
uniformly choose s_{i,j,0}, s_{i,j,1} \in \{0,1\}^n, set z_{i,j,0} = G(s_{i,j,0}), z_{i,j,1} = G(s_{i,j,1}),
     and pr_{i,j} = z_{i,j,0} \oplus z_{i,j,1}, for j = 1, \ldots, m;
set v_{i,j} = pr_{i,j} \oplus u_{i,j} and send v_{i,j} to V_i, for j = 1, \ldots, m;
get a message advmes from A;
if advmes is a decommitment message from V_i then
  if the decommitment is not appropriate then halt;
  if i=q then exit from the repeat loop else get a message advmes from A;
if advmes is a commitment message from a verifier \neq V_i then
  call such verifier V_{i+1} and set i = i + 1;
repeat
  get a message advmes from A;
  if advmes is a commitment message from a verifier \neq V_i then
     uniformly choose v_{i,1}, \ldots, v_{i,m} \in \{0, 1\}^{3n} and send them to \mathcal{A};
  if advmes is a decommitment message from a verifier \neq V_i then
     if the decommitment is not appropriate then halt;
until advmes is the decommitment message from V_i;
if the decommitment by V_i is not appropriate then halt;
let u_{i,1}, \ldots, u_{i,m} \in \{0,1\}^{3n} be the strings decommitted from V_1;
```

3. for i = 1, ..., m, let α_i be a transcript of the communication between Sim and V_i , let $\beta_i = (r_i \circ \alpha_i)$, where r_i is the random tape used by V_i and let T_1 be the entire transcript between all P_i 's and all V_i 's so far.

Instructions for Sim (proof phase):

```
1. Repeat
```

```
get a message advmes from A;
      if advmes = 'start i-th proof' then
         uniformly choose rand_i and let (mes_i, c_i, ans_i) = S(rand_i, x_i);
      send z_{i,1,0},\ldots,z_{i,m,0} to \mathcal{A};
      if advmes = b_i \in \{0,1\}^m then
         send ans_i and let d_{ij} be the j-th bit of c_i \oplus b_i;
         decommit each z_{i,j,0} as d_{ij} by sending to \mathcal{A} s_{i,j,0} if d_{ij} is 0 or s_{i,j,1} if d_{ij} is 1;
2. let T_2 be the entire transcript between all P_i's and all V_i's in this phase;
3. output: (T_1, T_2, \beta_1, \ldots, \beta_m).
```

Figure 1: The simulator algorithm Sim

of an execution of the computationally equivocable commitment scheme presented in Section 4. Therefore, we can directly apply Theorem 1 to conclude that the distribution of such triple in the real proof and the distribution of such triple in the simulation are computationally indistinguishable.

The second case we consider is the one in which the adversary \mathcal{A} decommits its commitments in step 1 in a proper way but decommits in a non proper way in one of the remaining steps of Sim. Notice that in this case Sim halts without output and therefore the two distributions differ. However we now see that this happens with negligible probability. Assume not; then \mathcal{A} is able to distinguish between the first simulated execution (in which all the $u_{i,j}$'s are random) and the second simulated execution (in which the $u_{i,j}$'s are pseudo-random). In the proof of Theorem 1 we showed that in the case of a single execution of a single bit protocol, this fact implies the existence of an efficient distinguisher between a random string and the output of G, thus contradicting the assumption that G is a pseudo-random generator. A modification of that proof can be used here too, in the concurrent setting, where the modification consists in using a hybrid argument to take care of the fact that there are several strings $u_{i,j}$ that are random in one execution and pseudo-random in the second execution.

The last case we need to consider is the one in which the adversary \mathcal{A} does not decommit in a proper way its commitments in step 1 of algorithm Sim. Note that if this happens the simulator Sim halts by outputting the transcript so far; this is the same view as that of \mathcal{A} during the real proof (since the prover also halts in this case); therefore, the simulation is perfect in this case.

5.2 Remarks and Extensions

We present some remarks and extensions of our main result, as given in Theorem 3, concerning minimal complexity assumptions, efficiency, allowing any polynomial number of proofs, and extending to any public-coin protocol.

Efficiency. In terms of communication and computation complexity, when applied to several 3-round public-coin honest-verifier zero-knowledge protocols in the literature (e.g., [17,5,18,25,6]) our proof system has efficiency comparable to that of the original protocol. In particular, contrarily to previously given concurrent zero-knowledge proof system, the construction of our protocol does not require any NP reduction, which could potentially blow up the parameters. Considering that our protocol is also a proof of knowledge, this may have applications to identification schemes. Moreover, in many examples for which the original computational zero-knowledge proof system already requires a commitment scheme (e.g., [17,5]), an optimization can be performed: such protocols can be made concurrent zero-knowledge in our model by only implementing the commitment scheme using our equivocable commitment scheme in Section 4.

Minimal complexity assumptions. Our protocol is based on the existence of perfectly secure commitment schemes and pseudo-random generators. Both can be constructed under the assumption of the existence of collision-intractable functions, and number-theoretic assumptions as the intractability of deciding quadratic residuosity modulo composite integers.

Any polynomial number of proofs. Our proposed protocol allows a number of proofs bounded by the size of the preprocessing. By combining our techniques with those in [13] for non-interactive zero-knowledge proofs, we can allow any polynomial (in the size of the preprocessing) number of proofs. This variation,

however, involves a general NP reduction, thus making the protocol much less efficient in terms of communication and computation complexity.

Extending to any public-coin honest-verifier zero-knowledge proof system. This extension is obtained by just observing that the transformation done for the challenge message in the 3-round protocol in the proof of Theorem 3 can be performed to all public-coin messages from the verifier. In particular, our results apply to all languages having an interactive proof system since they have a public-coin honest-verifier zero-knowledge proof system [2, 19].

6 Concurrent Zero-Knowledge Arguments with preprocessing

We now consider the problem of constructing concurrent perfect zero-knowledge arguments with preprocessing. Specifically, we show that the protocol in Section 5, when implemented using the perfectly equivocable commitment scheme in Section 4, allows to obtain concurrent perfect zero-knowledge arguments. Formally, we achieve the following

Theorem 4. Let L be a language in NP. Assuming the existence of perfectly secure commitment schemes, there exists (constructively) a concurrent perfect zero-knowledge argument with preprocessing for L where the preprocessing and proof phase require 3 rounds of communication, and the soundness property holds under the intractability of computing discrete logarithms modulo a prime.

The proof of the above theorem goes along the same lines as the proof of Theorem 3 and therefore we only sketch it, by pointing out the few differences. First of all, we reduce L to an NP-complete language, say, Hamiltonian Graphs. Then we can use the 3-round public-coin honest-verifier zero-knowledge proof system given in [5], call it (A,B). Starting from this protocol, we construct a protocol (P,V) as follows. Whenever, during the protocol (A,B), A is required to use a computationally equivocable commitment scheme, we will require A to run the perfectly equivocable commitment scheme in Section 4. The intuition here, also used in [14], is that the ability of decommitting each commitment to mes_i as any desired string allows to perform a perfect simulation of protocol (A,B) without need of a witness for the original input graph. We then have that the soundness holds under the same properties than the binding property of such commitment scheme, that is, the intractability of computing discrete logarithms modulo a prime. The concurrent zero-knowledge property is proved as for Theorem 3. We note that this technique also applies to languages having 3-round public-coin honest-verifier perfect zero-knowledge proof systems as those in [18, 17, 25, 6], in which case it returns a perfect zero-knowledge argument with preprocessing and with efficiency (in terms of communication and computational complexity) comparable to the starting protocol. Specifically, we do not need any NP reduction during the construction of the protocol, even if we want to run a number of proofs polynomial in the length of the preprocessing.

Acknowledgement. We thank Ran Canetti for useful discussions regarding equivocable commitments.

References

- D. Beaver, Adaptive Zero-Knowledge and Computational Equivocation, Proc. of FOCS 96.
- M. Ben-Or, J. Hastad, J. Kilian, O. Goldreich, S. Goldwasser, S. Micali, and P. Rogaway, Everything Provable is Provable in Zero-Knowledge, Proc. of CRYPTO 88.
- 3. T. Beth and Y. Desmedt, *Identification Tokens or: Solving the Chess Grandmaster Problem*, Proc. of CRYPTO 90.
- 4. G. Brassard, D. Chaum, and C. Crepeau, *Minimum Disclosure Proofs of Knowledge*, Journal of Computer and System Sciences, vol. 37, n. 2, 1988, pp. 156-189.
- 5. M. Blum, How to Prove a Theorem So No One Else Can Claim It, Proc. of the International Congress of Mathematicians, California, 1986, pp. 1444-1451.
- 6. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung, On Monotone Formula Closure of SZK, Proc. of FOCS 94.
- 7. A. De Santis, S. Micali, and G. Persiano, Non-Interactive Zero-Knowledge Proof Systems with Preprocessing, Proc. of CRYPTO 88.
- 8. G. Di Crescenzo, A Note on Distributed Zero-Knowledge Proofs, unpublished manuscript (submitted to PODC, January 1997).
- 9. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky, Non-Interactive and Non-Malleable Commitment, Proc. of STOC 98.
- 10. D. Dolev, C. Dwork, and M. Naor, *Non-Malleable Cryptography*, Proc. of STOC 91.
- C. Dwork, M. Naor, and A. Sahai, Concurrent Zero-Knowledge, Proc. of STOC 98.
- C. Dwork and A. Sahai, Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints, Proc. of CRYPTO 98.
- 13. U. Feige, D. Lapidot, and A. Shamir, Multiple Non-Interactive Zero-Knowledge Proofs based on a Single Random String, Proc. of FOCS 90.
- U. Feige and A. Shamir, Zero-Knowledge Proofs of Knowledge in Two Rounds, Proc. of CRYPTO 89.
- 15. O. Goldreich, Foundations of Cryptography: Fragments of a Book, in Electronic Colloquium on Computational Complexity.
- O. Goldreich and A. Kahan, How to construct constant-round zero-knowledge proof systems for NP, in Journal of Cryptology, vol. 9, n. 3, pp. 167-190.
- 17. O. Goldreich, S. Micali, and A. Wigderson, Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems Journal of the ACM, vol. 38, n. 1, 1991, pp. 691-729.
- 18. S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, February 1989.
- 19. R. Impagliazzo and M. Yung, Direct Minimum-Knowledge Computation, Proc. of CRYPTO 87.
- 20. J. Kilian, E. Petrank, and C. Rackoff, Lower Bounds for Zero-Knowledge on the Internet, Proc. of FOCS 98.
- 21. M. Naor, Bit Commitment from Pseudo-Randomness, Proc. of CRYPTO 91.
- 22. M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung, Perfect Zero-Knowledge Arguments under General Complexity Assumptions, Proc. of CRYPTO 92.
- 23. T. Pedersen, Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, Proc. of CRYPTO 91.
- 24. R. Richardson and J. Kilian, On the Concurrent Composition of Zero-Knowledge Proofs, Proc. of EUROCRYPT 99.
- 25. M. Tompa and H. Woll, Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information, Proc. of FOCS 87.

On the Security Properties of OAEP as an All-or-Nothing Transform*

Victor Boyko^{⋆⋆}

MIT Laboratory for Computer Science 545 Technology Square, Cambridge, MA 02139 boyko@theory.lcs.mit.edu

Abstract. This paper studies All-or-Nothing Transforms (AONTs), which have been proposed by Rivest as a mode of operation for block ciphers. An AONT is an unkeyed, invertible, randomized transformation, with the property that it is hard to invert unless all of the output is known. Applications of AONTs include improving the security and speed of encryption. We give several formal definitions of security for AONTs that are stronger and more suited to practical applications than the original definitions. We then prove that Optimal Asymmetric Encryption Padding (OAEP) satisfies these definitions (in the random oracle model). This is the first construction of an AONT that has been proven secure in the strong sense. Our bound on the adversary's advantage is nearly optimal, in the sense that no adversary can do substantially better against the OAEP than by exhaustive search. We also show that no AONT can achieve substantially better security than OAEP.

Key words: all-or-nothing transforms, encryption modes, OAEP, random oracles, polynomial indistinguishability, semantic security, exact security.

1 Introduction

The concept of an $All-or-Nothing\ Transform\ (AONT)$ was introduced by Rivest [18] to increase the cost of brute force attacks on block ciphers without changing the key length. As defined in that paper, an AONT is an efficiently computable transformation f, mapping sequences of blocks (i.e., fixed length strings) to sequences of blocks, which has the following properties:

- Given all of $f(x_1, \ldots, x_n) = (y_1, \ldots, y_{n'})$, it is easy to compute x_1, \ldots, x_n .
- Given all but one of the blocks of the output (i.e., given $y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_{n'}$ for any $1 \le j \le n'$), it is infeasible to find out any information about any of the original blocks x_i .

^{*} This version of the paper has been truncated due to page limits. The full version is available from the author's homepage [5].

^{**} Supported by an NSF Graduate Research Fellowship and DARPA grant DABT63-96-C-0018.

As mentioned by Rivest [18], an AONT should be randomized, so that a known message does not yield a known output.

An AONT itself does not perform any encryption, since there is no secret key information involved. However, if its output is encrypted, block by block, with a block cipher, the resulting scheme will have the following interesting property: An adversary cannot find out any information about any block of the message without decrypting all the blocks of the ciphertext. Now if the adversary attempts to do an exhaustive search for the key, she will need to perform n' decryptions before determining whether a given key is correct. Thus, the attack will be slowed down by a factor of n', without any change in the size of the secret key. This is particularly important in scenarios where the key length is constrained to be insecure or marginally secure (e.g., because of export regulations).

Another very important application of AONTs, as proposed by Johnson et al. [12] for inclusion in the IEEE P1363a standard, is to make fixed-blocksize encryption schemes more efficient. Instead of encrypting the whole message block by block, we can apply AONT to it, and encrypt just some of the blocks of the output. This will be an improvement if the AONT is more efficient than the cipher. It is especially useful if the cipher is a public key cryptosystem, such as RSA [17] or ElGamal [6]. This way we can, for instance, use RSA to securely encrypt messages longer than the key size, without need for a symmetric cipher. This gives an even greater improvement for elliptic-curve cryptosystems, which typically have a block length that is too small to efficiently use the traditional approach of encrypting a symmetric session key, together with padding and redundancy (see Johnson and Matyas [11]). A similar application of AONTs, as proposed by Rivest [19], would be to reduce communication requirements, in case the encryption function greatly expands its input.

The use of AONT with encryption can be particularly useful for remotely keyed encryption, i.e., applications where the part of the system that contains the keys is separate, and where bandwidth restrictions prevent us from sending the whole message from the insecure to the secure component [4]. An example of such a scenario would be the case where the keys are stored in a smartcard, and the user wishes to encrypt or decrypt large files. Through the use of AONT, we can completely eliminate any encryption components from the host system, and restrict such operations to the smart card (this is a generalization of the scheme of Jakobsson et al. [10], substituting general AONTs for the OAEP-like construction used in that paper). The host would transform the message with an AONT, and send one block to the smartcard. The smartcard would encrypt that block, and return it to the host. The encryption of the message will then be the output of the AONT, with one block encrypted. Assuming the block encryption is secure, the whole message will be secure. Note that since the host system does not contain any encryption algorithms, it might not be subject to export regulations.

The major problem with the definition of Rivest [18] is as follows: That definition only speaks about the amount of information that can be learned

about a particular message block. It does not, however, address the issue of information about the message as a whole (e.g., the XOR of all the blocks). To make the AONT truly useful, we would want it to hide all information about the input if any part of the output is missing (we will refer to this as the semantic security model). For instance, if an AONT is used for the purpose of slowing down exhaustive search of the key space, a relation between several blocks of the plaintext may provide enough information to the adversary for the purpose of detecting an invalid key.

Another disadvantage of Rivest's model [18] is that it does not consider the relation between the number of bits of AONT output that the adversary has, and the information that is leaked about the input. That model only considers the cases when the adversary has the whole output (in which case she should be able to completely determine the input), and when at least one complete block of the output is missing (in which case it should be infeasible to determine any block of the input). It would be interesting to consider exactly how much information about the input can be determined by looking at all but a certain number l bits of the AONT output, and how much effort is required to obtain that information.

1.1 This Work

The goal of this paper is to provide an AONT construction that is provably secure in the strong sense described above. Our contributions are as follows:

- We give new formal definitions of AONT security in terms of semantic security and indistinguishability. These definitions address the concerns mentioned above and provide the security needed for practical applications. They are parallel to the two notions of security for public-key cryptosystems, defined by Goldwasser and Micali [9]. We consider both the non-adaptive scenario (where the positions of the bits that are removed from AONT output are fixed before the experiment), and the adaptive scenario (where the adversary can choose the positions).
- We prove that OAEP (see Sect. 1.2), a construction originally introduced by Bellare and Rogaway in a different context, satisfies these definitions (in the random oracle model).
- We give an upper bound on the adversary's advantage in getting information about OAEP input when given all but l bits of OAEP output, as opposed to having none of the output. The bound is exact, i.e., does not involve asymptotics. It does not use any computational assumptions and relies only on the properties of random oracles. The bound is directly proportional to the number of adversary's queries to the random oracle and is inversely exponential in the number of bits of OAEP output that are withheld from the adversary.
- We then show that our upper bound is nearly optimal, in the sense that no adversary can do substantially better against OAEP than by exhaustive search. In addition, it will follow that no AONT can achieve substantially better security (i.e., upper bound on the adversary's advantage) than OAEP.

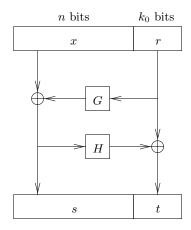


Fig. 1. A diagram of the OAEP

1.2 OAEP

Optimal Asymmetric Encryption Padding (OAEP) was originally introduced by Bellare and Rogaway [2] for the purpose of constructing semantically secure and plaintext-aware public-key encryption schemes from arbitrary trapdoor permutations. For parameters n and k_0 , "generator" $G: \{0,1\}^{k_0} \to \{0,1\}^n$, and "hash function" $H: \{0,1\}^n \to \{0,1\}^{k_0}$, the transform OAEP: $\{0,1\}^n \times \{0,1\}^{k_0} \to \{0,1\}^{n'}$, for $n' = n + k_0$, is defined as

$$OAEP^{G,H}(x,r) = x \oplus G(r) || r \oplus H(x \oplus G(r))$$
,

where \parallel denotes concatenation. Here x is the message and r is a random string. In applications, n would be the length of a message, and k_0 would be the security parameter, e.g., $k_0 = 128$. We will often refer to the first half of the OAEP output (i.e., $x \oplus G(r)$) as s, and to the second half (i.e., $r \oplus H(s)$) as t. Here |s| = n and $|t| = k_0$. We may also write OAEP^{G,H}(x), implying that r is chosen uniformly at random from $\{0,1\}^{k_0}$.

A diagram of the OAEP appears in Fig. 1.

Functions G and H are "random oracles," as introduced in by Bellare and Rogaway [1]. Bellare and Rogaway [2] show that if G and H are "ideal," i.e., they are random functions, and $f: \{0,1\}^{k_0+n} \to \{0,1\}^{k_0+n}$ is a trapdoor permutation, then the encryption scheme

$$\mathcal{E}^{G,H}(x) = f(OAEP^{G,H}(x,r))$$
,

with r chosen at random for each encryption, is semantically secure, in the sense of Goldwasser and Micali [9]. They also show that a small modification of that scheme provides plaintext-awareness (introduced by Bellare and Rogaway [2]).

1.3 Previous Work

Rivest [18] has proposed the following construction ("the package transform") as a candidate AONT:

- Let E be a block cipher. Let K_0 be a fixed, publicly known key for E.
- Let the input message be the sequence of blocks m_1, m_2, \ldots, m_s .
- Choose at random a key K' for E.
- Compute the output sequence $m'_1, m'_2, \ldots, m'_{s'}$, for s' = s + 1, as follows:
 - Let $m'_i = m_i \oplus E(K', i)$ for $i = 1, 2, \ldots, s$.
 - Let

$$m'_{s'} = K' \oplus h_1 \oplus h_2 \oplus \cdots \oplus h_s,$$

where

$$h_i = E(K_0, m_i' \oplus i)$$

for
$$i = 1, 2, ..., s$$
.

No formal proof was given that this construction is actually an AONT. The heuristic argument for security is based on the idea that if any block of the output is unknown, then K' cannot be computed, and so it is infeasible to compute any message block. Rivest [18] mentions that "the package transform" can be viewed as a special case of the OAEP, for $G(x) = E(x,1)||E(x,2)|| \cdots ||E(x,s)|$ and $H(x) = \bigoplus_{i=1}^{s} E(K_0, x_i \oplus i)$. However, no claims about OAEP itself are made in that paper.

Johnson et al. [12], in their contribution for the IEEE P1363a standard, give an OAEP-like transform that uses four rounds of hash applications instead of two. A heuristic analysis of the security of that construction is given by Matyas et al. [13]. Using an informal assumption about the hardness of the underlying hash functions, they argue that the number of operations required to determine the secret bits in the input message grows exponentially with the number of unknown bits. However, we are not aware of any formal proof of security of the transform of Johnson et al. [12]. In any case, the analysis of Matyas et al. [13] is not directly applicable if there are fewer than four rounds, so it does not work for OAEP.

Stinson [20] gives a treatment of AONTs from the point of view of unconditional security. Similarly to Rivest [18], Stinson's definition only considers the amount of information leaked about a particular block of the message, as opposed to the whole message. He uses a straightforward formalization of Rivest's definition above, suitably modified for information-theoretic security. Stinson then goes on to propose some constructions for AONTs using linear transforms, which can be proven secure in that model. The basic idea of these constructions is to use the function $\phi(\mathbf{x}) = \mathbf{x}M^{-1}$, where \mathbf{x} is a vector of s message blocks (considered as elements of GF(q), for some prime power q), and M is an invertible s by s matrix over GF(q), such that no entry of M is equal to 0. It is

easy to see that each component of \mathbf{x} linearly depends on all the components of $\mathbf{y} = \phi(\mathbf{x})$ (since $\mathbf{x} = \mathbf{y}M$).

It is conceivable that "the package transform" of Rivest [18] would be secure in the semantic security model (with sufficiently strong assumptions about the block cipher). The construction of Johnson et al. [12] may also be secure, although no formal proof has been given. However, the linear constructions of Stinson [20] would definitely not be secure in that model, since it is easy to come up with linear relations among the elements of \mathbf{x} by looking at just a few elements of $\phi(\mathbf{x})$ (in fact, since ϕ is linear and deterministic, every output of $\phi(\mathbf{x})$ gives a linear relation on elements of \mathbf{x}). Even if the message is padded with random blocks, it is still possible to extract partial information about the message if the number of known outputs is larger than the number of random blocks.

It is interesting to note that the relationship between the number of missing bits and adversary's required effort has come up in other contexts. Merkle [14], in one of the first papers on public key cryptography, defines the concept of a "puzzle," which is a cryptogram that requires $\Theta(N)$ work to break, where N is some number depending on the security parameters (the total amount of work put in by the communication parties is going to be $\Theta(N)$). Merkle's proposed construction of such "puzzles" is to take a block cipher and restrict the size of the key space, by varying only $\Theta(\log N)$ bits of the key and fixing the rest. It is assumed that breaking a cryptogram of the underlying cipher, when all but $\Theta(\log N)$ bits of the key are known, requires $\Theta(N)$ work.

Even et al. [7] assume the existence of a "uniformly secure" block cipher for their construction of a contract signing protocol. They consider a block cipher "uniformly secure" if it is infeasible to find a key for a given plaintext-ciphertext pair when no information about the key is known; but if the first i bits of the key are known, then there is an algorithm for breaking the cryptogram in time t(k-i), for some function $t(\cdot)$, and no algorithm can do it faster than in time $\frac{1}{2}t(k-i)$. Here k is the key length.

Both Merkle [14] and Even et al. [7] conjecture that standard block ciphers, such as Lucifer [8] or DES [16], satisfy their assumptions. However, uniform security is probably not a common consideration in block cipher design, as almost all applications of these primitives assume the whole key to be secret. Thus it may be unsafe to make such an assumption about standard block ciphers. In fact, this is, in effect, one of the criticisms given by Ben-Or et al. [3] of the work of Even et al. [7]. It seems to us, however, that the methods of our paper can be used to give a simple construction that will turn any block cipher that is secure in the regular sense into one which is uniformly secure. See Sect. 5 for details.

1.4 Outline

The outline of the rest of the paper is as follows. Section 2 describes the notation and model. In Sect. 3, we give formal definitions of security for AONTs. Section 4 presents the results on the security of OAEP as an AONT. Section 5 discusses open problems.

2 Notation and Model

Let us speak briefly about our notation and model. All algorithms used are oracle Turing machines, possibly randomized. Oracle queries execute in unit time. If A is a randomized algorithm, we may write $A(x_1, \ldots)$ to mean the distribution of A's output on certain inputs. We may also specify the coins explicitly, as in $A(r_A, x_1, \ldots)$, in which case the notation will refer to the fully determined output.

We will write $x \stackrel{R}{\leftarrow} X$ to mean that a variable x is to be chosen at random according to distribution X. As a shorthand, $x_1, x_2 \stackrel{R}{\leftarrow} X$ denotes $x_1 \stackrel{R}{\leftarrow} X$, $x_2 \stackrel{R}{\leftarrow} X$. On the other hand, $x \leftarrow X$ will mean that x is to be set to the result of evaluating expression X (which is not random). If S is a set, then we will write $x \stackrel{R}{\leftarrow} S$ to mean that x is chosen uniformly at random from S. We will write $\Pr[x \stackrel{R}{\leftarrow} X; y \stackrel{R}{\leftarrow} Y; z \leftarrow Z; \dots : p(x, y, z, \dots)]$ to mean the probability of predicate $p(x, y, z, \dots)$, when x is chosen at random according to distribution X, y is chosen at random according to distribution Y, z is set to the result of evaluating expression Z (possibly a function of x and y), etc. Similarly, we will write $E[x \stackrel{R}{\leftarrow} X; \dots : f(x, \dots)]$ to mean the expected value of $f(x, \dots)$ when x is chosen at random according to distribution X, etc.

To specify the distribution of a random function ("random oracle"), such as G and H for OAEP, we will use notation like $G, H \stackrel{R}{\leftarrow} \Omega$, where Ω is the set of all maps from the set $\{0,1\}^*$ of finite strings to the set $\{0,1\}^\infty$ of infinite strings. The notation should be interpreted as appropriate in its context, restricting the input and truncating the output of the function as necessary.

For $x \in \{0,1\}^*$, $1 \le i \le |x|$, and $0 \le l \le |x| - i + 1$, let $\operatorname{substr}(x,i,l)$ denote the substring of x starting at bit i (with the leftmost bit being 1) and having length l.

For any integer m and $L \subseteq [1, m]$, we define $h_{m,L} : \{0, 1\}^m \to \{0, 1\}^{m-|L|}$ as follows: $h_{m,L}$ takes a bit string of length m and throws out ("hides") the bit positions indicated by L. More precisely, if we let \bar{L}_i , for $1 \le i \le m - |L|$ denote the ith smallest element of $\bar{L} = [1, m] \setminus L$, then

$$h_{m,L}(x) = \operatorname{substr}(x, \bar{L}_1, 1) \| \operatorname{substr}(x, \bar{L}_2, 1) \| \cdots \| \operatorname{substr}(x, \bar{L}_{m-|L|}, 1)$$
.

For
$$n' \ge l \ge 0$$
, let $\binom{n'}{l} = \{L \subseteq [1, n'] : |L| = l\}$.

3 Definitions

Our definitions of security for AONTs are patterned after the notions of security for encryption defined by Goldwasser and Micali [9]: polynomial security (polynomial indistinguishability) and semantic security. We also try to define security "exactly," as in Bellare and Rogaway [2]: instead of concerning ourselves

¹ To prevent confusion, we note that while Bellare and Rogaway [2] talk about semantic security (for encryption), the definition they give is actually stated in terms

with asymptotics (i.e., showing that the adversary's advantage is "negligible" in the security parameters), we are interested in giving an exact bound on the adversary's advantage, as a function of the adversary's running time, the number of bits of AONT's output given to the adversary, etc.

For simplicity, we will formulate the definitions in terms of a single random oracle Γ . No generality is lost, since a single random oracle can be used to simulate several, by constructing the query as the concatenation of the oracle index and the original query. For instance, we could use Γ to simulate random oracles G and H by translating query x to G into query $0 \parallel x$ to Γ and query y to H into $1 \parallel y$. In addition, it would be easy to change the definitions for the case of no random oracles.

The non-adaptive indistinguishability scenario is as follows: Let L be an arbitrary set of l bit positions. The adversary runs in two stages:

- 1. **Find stage:** The adversary is given L and access to Γ . She outputs $x_0 \in \{0,1\}^n$, $x_1 \in \{0,1\}^n$, and $c_f \in \{0,1\}^*$.
- 2. **Guess stage:** The adversary is given c_f and, for random bit b, AONT $^{\Gamma}(x_b)$ with bit positions L missing. The adversary has access to Γ . She has to guess b.

Note that x_0 and x_1 do not need to be explicitly passed to the guess stage, since they may be included in c_f . We may view c_f as the saved state of the adversary at the end of the find stage.

We want the adversary's probability of correctly guessing b to be as close as possible to $\frac{1}{2}$. The formal definition is as follows:

Definition 1 (Non-adaptive indistinguishability). Let AONT be a randomized transform mapping n-bit messages to n'-bit outputs and using random oracle Γ . Let l be between 1 and n'. An adversary A is said to succeed in $(T, q_{\Gamma}, \epsilon)$ -distinguishing AONT with l missing bits if there exists $L \in \binom{n'}{l}$ such that

$$\begin{split} \Pr[\varGamma \xleftarrow{R} \varOmega; (x_0, x_1, c_{\mathsf{f}}) \xleftarrow{R} A^{\varGamma}(L, \mathsf{find}); b \xleftarrow{R} \{0, 1\}; \\ y \xleftarrow{R} \mathsf{AONT}^{\varGamma}(x_b) : A^{\varGamma}(h_{n', L}(y), c_{\mathsf{f}}, \mathsf{guess}) = b] \geq \frac{1}{2} + \epsilon \enspace , \end{split}$$

and, moreover, in the experiment above, A runs for at most T steps, and makes at most q_{Γ} queries to Γ .

It follows from this definition that in order for an AONT to be secure in the sense of non-adaptive indistinguishability for certain choices of parameters, it needs to be that for every adversary and every L, the adversary's advantage has to be less than ϵ .

of indistinguishability. This is acceptable in their context, since the two notions are known to be equivalent for encryption (see Micali et al. [15]). In our context, however, we state and analyze each one separately, since no equivalence has yet been proven.

The adaptive indistinguishability scenario is as follows: The adversary runs in three stages. The first stage chooses a value of L, while the last two stages are same as in the non-adaptive indistinguishability scenario. The adversary runs as follows:

- 1. Select stage: The adversary is given l and access to Γ . She selects l bit positions and outputs $L \in \binom{n'}{l}$ and $c_s \in \{0,1\}^*$. 2. **Find stage:** The adversary is given c_s and access to Γ . She outputs $x_0 \in$
- $\{0,1\}^n$, $x_1 \in \{0,1\}^n$, and $c_f \in \{0,1\}^*$.
- 3. Guess stage: The adversary is given c_f and, for random bit b, AONT $^{\Gamma}(x_b)$ with bit positions L missing. The adversary has access to Γ . She has to guess b.

Similarly to the remark about x_0 and x_1 above, we note that L does not need to be explicitly passed to the find and guess stages, since it may be included in $c_{\rm s}$, and then put into $c_{\rm f}$.

In the formal definition, we will assume that the adversary's select stage will always output a valid value of $L \in \binom{n'}{l}$ (this can be implemented by having a suitable encoding).

Definition 2 (Adaptive indistinguishability). Let AONT be a randomized transform mapping n-bit messages to n'-bit outputs and using random oracle Γ . Let l be between 1 and n'. An adversary A is said to succeed in $(T, q_{\Gamma}, \epsilon)$ adaptively-distinguishing AONT with l missing bits if

$$\begin{split} \Pr[\Gamma \xleftarrow{R} \Omega; (L, c_{\mathsf{s}}) \xleftarrow{R} A^{\Gamma}(l, \mathsf{select}); (x_0, x_1, c_{\mathsf{f}}) \xleftarrow{R} A^{\Gamma}(c_{\mathsf{s}}, \mathsf{find}); \\ b \xleftarrow{R} \{0, 1\}; y \xleftarrow{R} \mathsf{AONT}^{\Gamma}(x_b) : A^{\Gamma}(h_{n', L}(y), c_{\mathsf{f}}, \mathsf{guess}) = b] \geq \frac{1}{2} + \epsilon \enspace , \end{split}$$

and, moreover, in the experiment above, A runs for at most T steps, and makes at most q_{Γ} queries to Γ .

Note that for the application of speeding up encryption that was mentioned above in Sect. 1, it is sufficient for the AONT to be secure for a fixed choice of the missing part of the output (since the user decides which part will be encrypted). Thus, for that application, it is sufficient for the AONT to be secure in the non-adaptive scenario. However, when an AONT is used to increase the cost of exhaustive search, it needs to be secure in the adaptive scenario, since then the adversary has a choice of which blocks to decrypt.

For the adaptive and non-adaptive indistinguishability scenarios, we will assume, without loss of generality, that A never asks the same oracle query more than once (A can accomplish this by remembering the history of past queries; this history can be passed between stages through c_s and c_f).

The non-adaptive semantic security scenario is as follows: Let L be an arbitrary set of l bit positions and $f: \{0,1\}^n \to \{0,1\}^*$ be an arbitrary deterministic function. The adversary runs in two unconnected stages (each stage can be viewed as a separate algorithm):

- **Find stage:** The adversary is given L and access to Γ . She outputs $x \in \{0,1\}^n$.
- **Guess stage** (no data is passed from the find stage): The adversary is given L and $AONT^{\Gamma}(x)$ with bit positions L missing. The adversary has access to Γ . She has to guess f(x).

In the context of the traditional definition of semantic security for encryption, the adversary's find stage may be seen as the sampling algorithm for a distribution of messages, and the guess stage as the actual predicting algorithm. We want the adversary not to be able to do substantially better than always outputting the most probable value of f(x). The formal definition is as follows:

Definition 3 (Non-adaptive semantic security). Let AONT be a randomized transform mapping n-bit messages to n'-bit outputs and using random oracle Γ . Let l be between 1 and n'. Let $f: \{0,1\}^n \to \{0,1\}^*$ be any deterministic function. An adversary A is said to succeed in (T,q_{Γ},ϵ) -predicting f from AONT with l missing bits if there exists $L \in \binom{n'}{l}$ such that

$$\begin{split} \Pr[\varGamma \xleftarrow{R} \varOmega; x \xleftarrow{R} A^{\varGamma}(L, \mathsf{find}); y \xleftarrow{R} \mathsf{AONT}^{\varGamma}(x) : \\ A^{\varGamma}(L, h_{n',L}(y), \mathsf{guess}) = f(x)] \geq p_{A,f} + \epsilon \enspace, \quad (1) \end{split}$$

where

$$p_{A,f} = E[\varGamma \xleftarrow{R} \varOmega : \max_{z} \Pr[x \xleftarrow{R} A^{\varGamma}(L,\mathsf{find}) : f(x) = z]] \enspace ,$$

and, moreover, in the experiment (1), A runs for at most T steps, and makes at most q_{Γ} queries to Γ .

The expectation in the definition of $p_{A,f}$ is necessary to handle the possibility that the adversary may choose x to be a function of Γ (e.g., x could be set to the result of querying Γ on some fixed input). This would result in perfect prediction (both the find and guess stages can compute the same x), even though the output of the find stage will appear random, for random Γ . Thus, the quantity

$$\max_{\boldsymbol{x}} \Pr[\boldsymbol{\Gamma} \overset{R}{\leftarrow} \boldsymbol{\varOmega}; \boldsymbol{x} \overset{R}{\leftarrow} \boldsymbol{A}^{\boldsymbol{\Gamma}}(\boldsymbol{L}, \mathsf{find}) : f(\boldsymbol{x}) = \boldsymbol{z}]$$

could be much smaller than the adversary's success probability. However, for any fixed Γ , this adversary would always output the same x, so $p_{A,f} = 1$. Thus, this adversary's advantage ϵ will have to be zero.

In the semantic security scenario (both adaptive and non-adaptive), no information is passed between the adversary's find and guess stages, except $h_{n',L}(AONT^{\Gamma}(x))$ (otherwise, the find stage could simply pass the value of f(x)). We will therefore remove the assumption that A can't make the same query to Γ more than once. We will still assume, though, that all queries are unique within a single stage.

The adaptive semantic security scenario is same as the non-adaptive one, except for the addition of the select stage before the find stage, in which the adversary outputs L. The formal definition is as follows:

Definition 4 (Adaptive semantic security). Let AONT be a randomized transform mapping n-bit messages to n'-bit outputs and using random oracle Γ . Let l be between 1 and n'. Let $f: \{0,1\}^n \to \{0,1\}^*$ be any deterministic function. An adversary A is said to succeed in (T,q_{Γ},ϵ) -adaptively-predicting f from AONT with l missing bits if

$$\Pr[\Gamma \stackrel{R}{\leftarrow} \Omega; (L, c_{\mathsf{s}}) \stackrel{R}{\leftarrow} A^{\Gamma}(l, \mathsf{select}); x \stackrel{R}{\leftarrow} A^{\Gamma}(c_{\mathsf{s}}, \mathsf{find}); y \stackrel{R}{\leftarrow} \mathsf{AONT}^{\Gamma}(x) : A^{\Gamma}(h_{n',L}(y), c_{\mathsf{s}}, \mathsf{guess}) = f(x)] \ge p_{A,f} + \epsilon , \quad (2)$$

where

$$p_{A,f} = E[\varGamma \xleftarrow{R} \varOmega; (L, c_{\mathsf{s}}) \xleftarrow{R} A^{\varGamma}(l, \mathsf{select}) : \max_{\mathsf{x}} \Pr[x \xleftarrow{R} A^{\varGamma}(c_{\mathsf{s}}, \mathsf{find}) : f(x) = z]] \enspace ,$$

and, moreover, in the experiment (2), A runs for at most T steps, and makes at most q_{Γ} queries to Γ .

Note that since information may be passed from the select stage to the find and guess stages (through c_s), we can assume that no query from the select stage is repeated in any of the other stages. There is no danger in passing c_s to the guess stage, since c_s is generated before x is chosen (note that $p_{A,f}$ involves an expectation over c_s , so the adversary will not gain any advantage by choosing (x, f(x)) at the select stage and then passing it to the other stages).

4 Security Results

Throughout most of this section we will be using two random oracles G and H. As mentioned above, we can still use our definitions, since Γ could be used to simulate G and H. We will write $A^{G,H}$ in place of A^{Γ} . We will also use notation $(T, q_G, q_H, \epsilon) - \cdots$ (e.g., "an adversary (T, q_G, q_H, ϵ) -distinguishes") as a shorthand for $(T, q_G + q_H, \epsilon) - \cdots$, with the additional condition that at most q_G queries are made to G and at most q_H queries are made to H.

4.1 Non-adaptive Indistinguishability: Upper Bound

Theorem 1. Suppose $l \leq k_0$ and $k_0 \geq 14$. Suppose that there exists an adversary A that (T, q_G, q_H, ϵ) -distinguishes OAEP with l missing bits, where $q_G \leq 2^{k_0-1}$. Then

$$\epsilon \le 8q_{\rm G} \frac{k_0}{\log_2 k_0} 2^{-l} .$$

The proof has been omitted due to page limits and can be found in the full version of this paper [5]. The intuition behind the result is as follows: Let r_0 be the value of r that was used to generate $\tilde{y} = h_{n',L}(\text{OAEP}^{G,H}(x_b))$ in a particular experiment. Then, the adversary cannot find out any information about x_b unless

she queries G for $G(r_0)$ (since x_b only appears in $OAEP^{G,H}(x_b, r_0)$ as $x_b \oplus G(r_0)$). There are $\sim 2^l$ possible values of r_0 , corresponding to the 2^l values of y that are consistent with \tilde{y} . Thus we would expect the probability that any of the adversary's queries to G are equal to r_0 to be bounded by approximately $q_G 2^{-l}$. The complication is that there may be fewer than 2^l possible values of r_0 and that these values may not be equally probable, given \tilde{y} . These possible variations in probability cause the term $O(\frac{k_0}{\log k_0})$.

Note that this result, like all the others in this paper, does not use any computational assumptions and the bound is information theoretic, based on the properties of random oracles. In fact, the bound does not directly depend on T, the adversary's running time. It does, however, have implications for the running time, since $T \ge q_{\rm G} + q_{\rm H}$ (every oracle query takes unit time).

4.2 Non-adaptive Indistinguishability: Lower Bound

To see how good our upper bound is, let us try to give a lower bound on the adversary's advantage, by estimating the success of exhaustive search. This lower bound applies to any AONT.

Theorem 2. Let AONT be a randomized transform mapping n-bit messages to n'-bit outputs and using random oracle Γ . Let l be between 1 and n-3. Then, for any $L \in {n' \choose l}$ and any N between 1 and 2^l , there exists an adversary that $(NT, Nq_{\Gamma}, \epsilon)$ -distinguishes AONT with l missing bits, with

$$\epsilon \ge \frac{1}{16} N 2^{-l}.$$

Here T and q_{Γ} are the time and number of queries to Γ , respectively, taken by a single evaluation of AONT.

The proof has been omitted due to page limits and can be found in the full version of this paper [5]. The idea of the proof is as follows: The exhaustive search algorithm that achieves the advantage of at least $\frac{1}{16}N2^{-l}$ works by choosing x_0 and x_1 independently at random in the find stage. The guess stage tries random values of the missing bits, up to N times, and, if the inverse AONT returns $x_{b'}$ for $b' \in \{0, 1\}$, produces b' as the guess. If none of the trials has succeeded, a random bit is returned. The idea of the analysis of this algorithm is that every trial in the guess stage has probability of at least 2^{-l} of succeeding with the correct value of b (since there exists a choice of the missing bits, namely the values that actually appeared in y, that leads to x_b). On the other hand, since x_{1-b} is chosen uniformly and independently, the probability of getting x_{1-b} in any particular trial is $2^{-n} \leq 2^{-l-3}$.

We see from Theorems 1 and 2, that no adversary can improve by a factor of more than $O(\frac{k_0}{\log k_0})$ over exhaustive search. Since, for large l, this factor is negligible compared to 2^{-l} , our bounds for OAEP are nearly optimal.

We also see that no AONT can be substantially more secure than OAEP, in the sense that no AONT can have an upper bound that is better than OAEP's by a factor of more than $O(\frac{k_0}{\log k_0})$.

4.3 Adaptive Indistinguishability

Theorem 3. Suppose $l \leq k_0$, $l \leq \frac{n}{2}$, and $k_0 \geq 14$. Suppose that there exists an adversary A that (T, q_G, q_H, ϵ) -adaptively-distinguishes OAEP with l missing bits, where $q_G \leq 2^{k_0-1}$. Then

$$\epsilon \le 8(q_{\rm G} + q_{\rm H}) \frac{k_0}{\log_2 k_0} 2^{-l}$$
.

The proof has been omitted due to page limits and can be found in the full version of this paper [5]. It is very similar to the proof of Theorem 1, with the only major difference being that we have to consider the possible correlation between L and H (since L may be chosen by the adversary to depend on H). This is taken care of by showing that with large probability (depending on q_H), the queries made in the select stage will not constrain H enough to spoil those properties of it that are used in the proof of Theorem 1.

We can easily see that for the adaptive indistinguishability scenario, as for the non-adaptive one, our bound is optimal within a factor of $O(\frac{k_0}{\log k_0})$ of the advantage given by exhaustive search for an arbitrary AONT (in this scenario, exhaustive search would choose a random L in the select stage).

4.4 Non-adaptive Semantic Security

Theorem 4. Suppose $l \le k_0$ and $k_0 \ge 14$. Suppose that there exists a deterministic function $f: \{0,1\}^n \to \{0,1\}^*$ and an adversary A that (T, q_G, q_H, ϵ) -predicts f from OAEP with l missing bits, where $q_G \le 2^{k_0-1}$. Then

$$\epsilon \le 8q_{\rm G} \frac{k_0}{\log_2 k_0} 2^{-l} .$$

The proof has been omitted due to page limits and can be found in the full version of this paper [5]. It is a simple modification of the proof of Theorem 1, with the difference being the estimation of the adversary's success probability in the case where she has not queried G for $G(r_0)$ (where r_0 is the value of r used to compute $OAEP^{G,H}(x)$ in the experiment). In the case of Theorem 1 that success probability was $\frac{1}{2}$, while here it can be easily seen to be less than or equal to $p_{A,f}$.

We have not yet shown a lower bound for the semantic security scenarios. We still expect our upper bound for these scenarios to be nearly optimal, as for the indistinguishability scenarios.

4.5 Adaptive Semantic Security

Theorem 5. Suppose $l \leq k_0$, $l \leq \frac{n}{2}$, and $k_0 \geq 14$. Suppose that there exists a deterministic function $f: \{0,1\}^n \to \{0,1\}^*$ and an adversary A that (T,q_G,q_H,ϵ) -adaptively-predicts f from OAEP with l missing bits, where $q_G \leq 2^{k_0-1}$. Then

$$\epsilon \le 8(q_{\rm G} + q_{\rm H}) \frac{k_0}{\log_2 k_0} 2^{-l}$$
.

The proof of Theorem 5 is a simple combination of the proof of Theorem 1 with the modifications needed for Theorems 3 and 4 (see the full version of this paper for details [5]).

5 Open Problems

The first open problem that comes to mind is to improve our bounds. The best would be to bring the upper bounds within a constant factor of exhaustive search, or to devise an algorithm that does better than exhaustive search. Also, it would be interesting to give lower bounds for the semantic security scenarios.

Another open problem is to show equivalence (or non-equivalence) of our definitions, trying to carry over the exact bounds as much as possible. There are also other possible models to consider, such as a scenario where, instead of specifying the positions of the missing bits in advance, the adversary is allowed to ask for bits after seeing the value of other bits. Also, just as there are several variations of the definition of semantic security for encryption, one might consider other definitions for AONTs, and whether they are equivalent to the ones in this paper.

One of the most interesting open problems related to AONTs is to construct a secure AONT (in the sense of our definitions) without the use of random oracles. One might start by trying to modify the OAEP by replacing either G or H by a deterministic function. In any case, until formal results about deterministic AONTs are obtained, it would be fruitful to investigate the practical security concerns that arise in the use of OAEP as an AONT, when the random oracles are instantiated with deterministic hash functions.

Another interesting question is whether there is any relation between the properties of OAEP as an AONT, and its original proposed use for constructing secure cryptosystems. One might ask, for instance, if OAEP could be replaced by an arbitrary AONT in the construction of Bellare and Rogaway [2].

One could also look into the possibility of generalizing the definitions of AONT security, so that instead of getting a certain number of bits of the output, the adversary gets the equivalent amount of information through other means, i.e., by seeing the value of some transformation of the output that reduces its entropy by l. The function $h_{n',L}$ is just one example of such a transformation.

Finally, now that we have a provably secure AONT, it would be of great interest to find new applications of this primitive. We hope that its usefulness extends far beyond its original applications.

We note that an AONT that satisfies our definitions, such as OAEP, can be used to implement the "puzzles" of Merkle [14], or the notion of uniform security of Even et al. [7]. The "puzzles" could be made by publishing a certain number of bits of AONT output on a bit string of sufficient redundancy (so that, with overwhelming probability, only one such string corresponded to the published information). Similarly, cryptograms of uniform security could be achieved by publishing a part of AONT output on the key used in the cryptogram. It seems, though, that a simpler construction would suffice: Let $E(K, M) \to C$ be a regular

symmetric cryptosystem, and let H be a random oracle. Then, it seems to us that, using the methods of this paper, it can be shown that $E(H(K), M) \to C$ is a uniformly secure cryptosystem. On the other hand, the AONT construction for the "puzzles" has the advantage that it does not use encryption, which could put it outside the scope of export regulations. It would be interesting to investigate these issues further.

Acknowledgments

I would like to express my deepest gratitude to Ronald Rivest for his direction, support, and frequent discussions. This work would have been impossible without him. I am also greatly indebted to Shafi Goldwasser for her numerous comments on the manuscript, helpful advice, and discussions. In addition, I would like to thank the following for their help, comments, and advice: Yevgeniy Dodis, Markus Jakobsson, Burt Kaliski, Daniele Micciancio, David Molnar, Kazuo Ohta, Alexander Perlin, Zulfikar Ramzan, Leo Reyzin, Phillip Rogaway, Julien Stern, and Yiannis Tsiounis.

References

- [1] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First Annual Conference on Computer and Communications Security*. ACM, 1993. Latest version can be obtained from http://www-cse.ucsd.edu/users/mihir/papers/ro.html.
- [2] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, Advances in Cryptology—EUROCRYPT 94, volume 950 of Lecture Notes in Computer Science, pages 92–111. Springer-Verlag, 1995, 9–12 May 1994. Latest version can be obtained from http://www-cse.ucsd.edu/users/mihir/papers/pke.html.
- [3] Michael Ben-Or, Oded Goldreich, Silvio Micali, and Ronald L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.
- [4] Matt Blaze. High-bandwidth encryption with low-bandwidth smartcards. In Dieter Grollman, editor, Fast Software Encryption: Third International Workshop, volume 1039 of Lecture Notes in Computer Science, pages 33–40, Cambridge, UK, 21–23 February 1996. Springer-Verlag.
- [5] Victor Boyko. On the security properties of OAEP as an all-or-nothing transform. Available from http://theory.lcs.mit.edu/~boyko/aont-oaep.html. Full version of the current paper.
- [6] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.
- [7] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. Communications of the ACM, 28:637–647, 1985.
- [8] H. Feistel. Cryptography and computer privacy. Scientific American, 228:15–23, May 1973.
- [9] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. Journal of Computer and System Sciences, 28(2):270–299, April 1984.

- [10] Markus Jakobsson, Julien Stern, and Moti Yung. Scramble all, encrypt small. In Fast Software Encryption: 6th International Workshop, Rome, Italy, 24–26 March 1999.
- [11] Don Johnson and Stephen Matyas. Asymmetric encryption: Evolution and enhancements. CryptoBytes, 2(1):1-6, Spring 1996. Available from ftp://ftp.rsa.com/pub/cryptobytes/crypto2n1.pdf.
- [12] Don Johnson, Stephen Matyas, and Mohammad Peyravian. Encryption of long blocks using a short-block encryption procedure. Available from http://grouper.ieee.org/groups/1363/contributions/peyrav.ps, November 1996. Submitted for inclusion in the IEEE P1363a standard.
- [13] Stephen Matyas, Mohammad Peyravian, and Allen Roginsky. Security analysis of Feistel ladder formatting procedure. Available from http://grouper.ieee.org/groups/1363/contributions/peyrav2.ps, March 1997.
- [14] R. C. Merkle. Secure communications over insecure channels. Communications of the ACM, 21:294–299, April 1978.
- [15] S. Micali, C. Rackoff, and R. H. Sloan. The notion of security for probabilistic cryptosystems. SIAM J. Computing, 17(2):412–426, April 1988.
- [16] National Institute of Standards and Technology (NIST). FIPS Publication 46: Announcing the Data Encryption Standard, January 1977. Originally issued by National Bureau of Standards.
- [17] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [18] Ronald Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, Fast Software Encryption: 4th International Workshop, volume 1267 of Lecture Notes in Computer Science, pages 210-218, Haifa, Israel, 20-22 January 1997. Springer-Verlag. Also available from http://theory.lcs.mit.edu/~rivest/fusion.ps.
- [19] Ronald Rivest. Chaffing and winnowing: Confidentiality without encryption. Available from http://theory.lcs.mit.edu/~rivest/chaffing-980701.txt, July 1998.
- [20] D.R. Stinson. Some observations on all-or-nothing transforms. Available from http://cacr.math.uwaterloo.ca/~dstinson/papers/AON.ps, 31 August 1998.

Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization

Mihir Bellare 1 and Amit Sahai 2

Abstract. We prove the equivalence of two definitions of non-malleable encryption appearing in the literature—the original one of Dolev, Dwork and Naor and the later one of Bellare, Desai, Pointcheval and Rogaway. The equivalence relies on a new characterization of non-malleable encryption in terms of the standard notion of indistinguishability of Goldwasser and Micali. We show that non-malleability is equivalent to indistinguishability under a "parallel chosen ciphertext attack," this being a new kind of chosen ciphertext attack we introduce, in which the adversary's decryption queries are not allowed to depend on answers to previous queries, but must be made all at once. This characterization simplifies both the notion of non-malleable encryption and its usage, and enables one to see more easily how it compares with other notions of encryption. The results here apply to non-malleable encryption under any form of attack, whether chosen-plaintext, chosen-ciphertext, or adaptive chosen-ciphertext.

1 Introduction

Public-key encryption has several goals in terms of protecting the data that is encrypted. The most basic is *privacy*, where the goal is to ensure that an attacker does not learn any useful information about the data from the ciphertext. Goldwasser and Micali's notion of indistinguishability [8] forms the accepted formalization of this goal. A second goal, introduced by Dolev, Dwork and Naor [5], is non-malleability, which, roughly, requires that an attacker given a challenge ciphertext be unable to modify it into another, different ciphertext in such a way that the plaintexts underlying the two ciphertexts are "meaningfully related" to each other. Both these goals can be considered under attacks of increasing severity: chosen-plaintext attacks, and two kinds of chosen ciphertext attacks [11,12].

Recent uses of public-key encryption have seen a growing need for, and hence attention to, stronger than basic forms of security, like non-malleability. This kind of security is important when encryption is used as a primitive in the design

Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: mihir@cs.ucsd.edu.

URL: http://www-cse.ucsd.edu/users/mihir.

MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139, USA. E-Mail: amits@theory.lcs.mit.edu.

of higher level protocols, for example for key distribution (cf. [1]) or electronic payment (cf. [13]). The interest is witnessed by attention to classification of the notions of encryption [2] and new efficient constructions of non-malleable schemes [3, 4].

In our discussions below, we begin for simplicity by focusing on the case where the notions are considered under chosen-plaintext attacks. We will discuss the extensions to stronger attacks later.

1.1 Themes in Foundations of Encryption

EQUIVALENCES. We said above that indistinguishability was the "accepted" formalization of the privacy goal. This agreement is due in large part to a body of work that has established that numerous other formalizations put forth to capture privacy are actually equivalent to indistinguishability. In particular this is true of semantic security [8] and for a notion of privacy based on computational entropy [14, 10]. These foundational results have since been refined and extended to other settings [7]. These equivalences are a cornerstone of our understating of privacy, providing evidence that we have in fact found the "right" formalization.

Characterizations. Semantic security captures in perhaps the most direct way one's intuition of a good notion of privacy. (Roughly, it says that "whatever can be efficiently computed about a message given the ciphertext can be computed without the ciphertext"). But it is a relatively complex and subtle notion to formalize. For this reason, it is somewhat difficult to use in applications of encryption. Indistinguishability has the opposite attributes. The formalization is simple, appealing and easy to use. (It says that if we take two equal-length messages m_0, m_1 , an adversary given an encryption of a random one of them cannot tell which it was with a probability significantly better than that of guessing). It is by far the first choice when analyzing the security of an encryption based application. But it is less clear (by just a direct examination of the definition) that it really captures an intuitively strong notion of privacy. However, we know it does, because it is in fact equivalent to semantic security. Accordingly, we can view indistinguishability as a "characterization" of semantic security, a simple, easy to use notion backed by the fact of being equivalent to the more naturally intuitive one.

Thus, beyond equivalences between notions, one also seeks a characterization that is simple and easy to work with, a role which for privacy is played by the notion of indistinguishability.

1.2 Questions for Non-malleability

The foundations of non-malleable encryption are currently not as well laid as those of privacy, for several reasons.

First, there are in the literature two different formalizations to capture the notion of non-malleable encryption. The first is the original definition of Dolev, Dwork and Naor [5], which we call SNM (simulation based non-malleability).

A second, somewhat simpler formalization was introduced by Bellare, Desai, Pointcheval and Rogaway [2], and we call it CNM (comparison based non-malleability). A priori, at least, the two seem to have important differences.

Second, there is no simple characterization of non-malleable encryption akin to indistinguishability for privacy. Rather, the current formalizations of non-malleability follow the definitional paradigm of semantic security and in particular both existing formulations are quite complex (even though that of [2] is somewhat simpler than that of [5]), and subtle at the level of details. A consequence is that non-malleability can be hard to use. The applicability of non-malleability would be increased by having some simple characterization of the notion.

Although not required for the statement of our results, it may be instructive to try to convey some rough idea of the existing formalizations. (Formal definitions of both notions can be found in Section 3.) The definitions involve considering some relation R between plaintexts, having an adversary output a distribution on some set of messages, and then setting up a challenge-response game. The adversary is given as input a ciphertext y of a plaintext x drawn from the message distribution, and must produce a vector of ciphertexts \mathbf{y} . If \mathbf{x} is the plaintext vector corresponding to y, security requires, roughly, that the adversary's ability to make $R(x, \mathbf{x})$ true in this game is not much more than her ability to make it true had she had to produce y without being given y at all, namely given no information about x other than its distribution. The two known definitions differ in how exactly they measure the success in the last part of the game. The SNM notion is based on the simulation paradigm: a scheme is secure if for any adversary there exists a simulator which does almost as well without any information about the challenge ciphertext given to the adversary. In the CNM formalization, there is no simulator: it is required instead that the success probability of the adversary under the "real" challenge and a "fake" challenge be about the same. Besides the fundamental difference of one being simulation based and the other not, the SNM notion does not allow the simulator access to the decryption oracle even when the adversary gets it, while the CNM notion allows the adversary access to the decryption oracle in both games. These differences would raise a priori doubts about whether the two notions could be equivalent. In particular, SNM would appear to be stronger since it asks for simulation even without access to decryption ability.

1.3 The Equivalence

In this paper we show that the above two notions of non-malleability are in fact equivalent. (This holds under all three kinds of attack mentioned above.) In other words, if a particular encryption scheme meets the SNM notion of security, then it also meets the CNM notion, and vice versa.

Thus, we are saying that two formalizations of non-malleability, that were originally proposed with somewhat different intuitions behind them, are in fact capturing the same underlying notion. Like the equivalences amongst notions of

privacy, this result serves to strengthen our conviction that this single, unified notion of non-malleability is in fact the appropriate one.

1.4 An Indistinguishability Based Characterization

Perhaps more interesting than the above-mentioned equivalence is a result used to prove it. This is a new characterization of non-malleability that we feel simplifies the notion and increases our understanding of it and its relation to the more classic notions. Roughly speaking, we show that non-malleability is actually just a form of indistinguishability, but under a certain special type of chosen-ciphertext attack that we introduce and call a parallel chosen-ciphertext attack. Thus, what appears to be a different adversarial goal (namely, the ability to modify a ciphertext in such a way as to create relations between the underlying plaintexts) corresponds actually to the standard goal of privacy, as long as we add power to the attack model. This represents a tradeoff between goals and attacks.

Our characterization dispenses with the relation R and the message space: it is just about a game involving two messages.

To illustrate, consider non-malleability under chosen-plaintext attack. Our characterization says this is equivalent to indistinguishability under a chosenciphertext attack in which the adversary gets to ask a single vector query of the decryption oracle. This means it specifies a sequence $\mathbf{c}[1], \dots, \mathbf{c}[n]$ of ciphertexts, and obtains the corresponding plaintexts $\mathbf{p}[1], \dots, \mathbf{p}[n]$ from the oracle. But the choice of $\mathbf{c}[2]$ is not allowed to depend on the answer to $\mathbf{c}[1]$, and so on. (So we can think of all the queries as made in parallel, hence the name. Perhaps a better name would have been non-adaptive queries, but the term non-adaptive is already in use in another way in this area and was best avoided.) This query is allowed to be a function of the challenge ciphertext. We call this type of attack a "parallel chosen-ciphertext attack". In more detail the game is that we take two equal-length messages m_0, m_1 , give the adversary a ciphertext y of a random one of them, and now allow it a single parallel vector decryption oracle query, the only constraint on which is that the query not contain y in any component. The adversary wins if it can then say which of the two messages m_0, m_1 had been encrypted to produce the challenge y, with a probability significantly better than that of guessing. Thus, as mentioned above, our notion makes no mention of a relation R or a probability space on messages, let alone of a simulator. Instead, it follows an entirely standard paradigm, the only twist being the nature of the attack model.

A special case that might be worth noting is that when the relation R is binary, the parallel attack need contain just one ciphertext. In general, the number of parallel queries needed is one less than the arity of R.

Recall that non-malleability at first glance is capturing a very different sort of notion than indistinguishability: the ability to modify a ciphertext in some way so as to create plaintext dependencies. This can be done even if one doesn't "know" the plaintexts involved, so it seems different from privacy. Our characterization brings out the fact that the difference is not so great: the increased severity of

the goal of non-malleability can be compensated for by a strong attack under indistinguishability.

1.5 Extensions and Discussion

We have focused above mostly on non-malleability under chosen-plaintext attack. Let us briefly discuss the extensions of the results to the two kinds of chosen-ciphertext attack. They are referred to in [2] as CCA1 and CCA2, and correspond to lunch-time [11] and adaptive [12] attacks.

As mentioned above, our equivalence result extends to the stronger attack forms. However, for the case of the strongest type of attack, namely CCA2, the result is not really a novelty, because it can be derived as a consequence of the results of [2] and [6]. (Specifically each of these showed that their notion of non-malleability under CCA2 is equivalent to indistinguishability under CCA2, so this makes the two notions of non-malleability under CCA2 equivalent to each other.) Thus the interest of our results is largely for the case of chosen-plaintext attack and CCA1.

A similar situation holds with regard to the characterization. It is simple to provide an appropriate extension of indistinguishability under parallel attack to the CCA1 and CCA2 settings, and we can show the characterization holds. But adding a parallel attack to CCA2 leaves the latter unchanged, and our results just collapses to the already know equivalence between non-malleability and indistinguishability in the CCA2 case.

1.6 Relations among Notions of Security

Our new characterization of non-malleability as indistinguishability under a parallel chosen-ciphertext attack simplifies the discussion of relationships among the notions of security studied in [2,6]. Our characterization shows that non-malleability with respect to chosen-plaintext attack, lunchtime chosen-ciphertext attack, or adaptive chosen-ciphertext attack is equivalent to indistinguishability with respect to a parallel chosen-ciphertext attack (denoted PA0), a lunchtime attack followed by a parallel attack (denoted PA1), or an adaptive chosen ciphertext attack (denote CCA2), respectively. Thus, by our equivalence, the six standard notions of security are equivalent to indistinguishability with respect to five different types of attack, denoted CPA, PA0, CCA1, PA1, and CCA2.

Now, to show that these notions are all distinct, it suffices to show for each pair of notions that an encryption scheme secure against the weaker form of attack can be modified to fail against the stronger attack, but still be secure against the weaker form of attack. Using our new characterization one can in fact separate all these notions by simply following the following guidelines: To make a system insecure under a lunchtime attack, one simply needs to fix a particular ciphertext which decrypts to the secret key. In order to require that a non-parallel attack be used to do this, one may add a level of indirection by having a particular ciphertext that decrypts to another randomly chosen ciphertext, whose decryption is the secret key. To modify a system that is secure

against lunchtime attacks in order to make it fail against an adaptive parallel attack, one simply establishes a rule that a ciphertext can be modified in a canonical manner to produce a new ciphertext that decrypts to the same value. Again, to require that a non-parallel attack be used, one may again add a level of indirection, by having the modified ciphertext decrypt to another ciphertext, which decrypts to the original message. These intuitions, which are very natural given the attacks, can readily be made into proofs which are simpler than those given by [2] to accomplish this goal.

Note also that each of [2] and [6] established relations using their own definitions of non-malleability. Their results are unified by our results showing the two notions of non-malleability are the same.

Above, we only consider parallel chosen-ciphertext attacks in the adaptive stage of the attack, *i.e.* when the adversary has seen the challenge ciphertext. However, it is quite natural to consider a parallel attack as also possible in the first stage of the attack, before the challenge ciphertext is known. This leads naturally to nine different notions of security against chosen-ciphertext attacks. It could be interesting to investigate these attacks and any relations which may exist among them.

1.7 Related Work

Halevi and Krawczyk introduce a weak version of chosen-ciphertext attack which they call a one-ciphertext *verification* attack [9]. This is not the same as a parallel attack. In their attack, the adversary generates a single plaintext along with a candidate ciphertext, and is allowed to ask a verification query, namely whether or not the pair is valid. In our notion, the adversary has more power: it can access the decryption oracle. No equivalences are proved in [9].

2 Preliminaries

EXPERIMENTS. We use standard notations and conventions for writing probabilistic algorithms and experiments. If A is a probabilistic algorithm, then $A(x_1, x_2, \ldots; r)$ is the result of running A on inputs x_1, x_2, \ldots and coins r. We let $y \leftarrow A(x_1, x_2, \ldots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \ldots; r)$. If S is a finite set then $x \leftarrow S$ is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. We say that y can be output by $A(x_1, x_2, \ldots)$ if there is some r such that $A(x_1, x_2, \ldots; r) = y$. Also, to clarify complicated probabilistic statements, we define random variables as experiments.

SYNTAX AND CONVENTIONS. The syntax of an encryption scheme specifies what kinds of algorithms make it up. Formally, an asymmetric encryption scheme is given by a triple of algorithms, $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where

• \mathcal{K} , the key generation algorithm, is a probabilistic algorithm that takes a security parameter $k \in \mathbb{N}$ (provided in unary) and returns a pair (pk, sk) of matching public and secret keys.

- \mathcal{E} , the *encryption algorithm*, is a probabilistic algorithm that takes a public key pk and a message $x \in \{0, 1\}^*$ to produce a ciphertext y.
- \mathcal{D} , the decryption algorithm, is a deterministic algorithm which takes a secret key sk and ciphertext y to produce either a message $x \in \{0,1\}^*$ or a special symbol \perp to indicate that the ciphertext was invalid.

We require that for all (pk, sk) which can be output by $\mathcal{K}(1^k)$, for all $x \in \{0, 1\}^*$, and for all y that can be output by $\mathcal{E}_{pk}(x)$, we have that $\mathcal{D}_{sk}(y) = x$. We also require that \mathcal{K} , \mathcal{E} and \mathcal{D} can be computed in polynomial time. As the notation indicates, the keys are indicated as subscripts to the algorithms.

Recall that a function $\epsilon: \mathbb{N} \to \mathbf{R}$ is negligible if for every constant $c \geq 0$ there exists an integer k_c such that $\epsilon(k) \leq k^{-c}$ for all $k \geq k_c$.

NOTATION. We will need to discuss vectors of plaintexts or ciphertexts. A vector is denoted in boldface, as in \mathbf{x} . We denote by $|\mathbf{x}|$ the number of components in \mathbf{x} , and by $\mathbf{x}[i]$ the i-th component, so that $\mathbf{x} = (\mathbf{x}[1], \dots, \mathbf{x}[|\mathbf{x}|])$. We extend the set membership notation to vectors, writing $x \in \mathbf{x}$ or $x \notin \mathbf{x}$ to mean, respectively, that x is in or is not in the set $\{\mathbf{x}[i]: 1 \leq i \leq |\mathbf{x}|\}$. It will be convenient to extend the decryption notation to vectors with the understanding that operations are performed component-wise. Thus $\mathbf{x} \leftarrow \mathcal{D}_{sk}(\mathbf{y})$ is shorthand for the following: for $1 \leq i \leq |\mathbf{y}|$ do $\mathbf{x}[i] \leftarrow \mathcal{D}_{sk}(\mathbf{y}[i])$.

We will consider relations of arity t where t will be polynomial in the security parameter k. Rather than writing $R(x_1, \ldots, x_t)$ we write $R(x, \mathbf{x})$, meaning the first argument is special and the rest are bunched into a vector \mathbf{x} with $|\mathbf{x}| = t - 1$.

3 Two Definitions of Non-malleable Encryption

In the setting of non-malleable encryption, the goal of an adversary, given a ciphertext y, is not (as with indistinguishability) to learn something about its plaintext x, but rather to output a vector \mathbf{y} of ciphertexts whose decryption \mathbf{x} is "meaningfully related" to x, meaning that $R(x,\mathbf{x})$ holds for some relation R. The question is how exactly one measures the advantage of the adversary. This turns out to need care. One possible formalization is that of [5,6], which is based on the idea of simulation; it asks that for every adversary there exists a certain type of "simulator" that does just as well as the adversary but without being given y. In another, somewhat simpler formalization introduced in [2], there is no simulator; security is defined via an experiment involving the adversary alone. We begin below by presenting these two notions.

3.1 Definition of SNM

In this subsection we describe the definition of non-malleable encryption of [5, 6], which we call SNM for "simulation based non-malleability." The SNM formulation fixes a polynomial time computable relation R, which is viewed as taking four arguments, $R(x, \mathbf{x}, M, s_1)$, with \mathbf{x} being a vector with an arbitrary number of components.

The adversary $A = (A_1, A_2)$ runs in two stages. The first stage of the adversary, namely A_1 , computes (the encoding of) a distribution M on messages and also some state information: a string s_1 to pass to the relation R, and a string s_2 to pass on to A_2 . (At A_1 's discretion, either of these might include M and pk.) We call M the message space. It must be valid, which means that all strings having non-zero probability under M are of the same length.

The second stage of the adversary, namely A_2 , receives s_2 and the encryption y of a random message x drawn from M. Algorithm A_2 then outputs a vector of ciphertexts \mathbf{y} . We say that A is successful if $R(x, \mathbf{x}, M, s_1)$ holds, and also $y \notin \mathbf{y}$ and $\bot \notin \mathbf{x}$, where $\mathbf{x} = \mathcal{D}_{sk}(\mathbf{y})$.

The requirement for security is that for any polynomial time adversary A and any polynomial time relation R there exists a polynomial time algorithm $S = (S_1, S_2)$, the simulator, that, without being given y, has about the same success probability as A. The experiment here is that S_1 is first run on pk to produce M, s_1, s_2 , then x is selected from M, then S_2 is run on s_2 (but no encryption of x) to produce \mathbf{y} . Success means $\mathbf{x} = \mathcal{D}_{sk}(\mathbf{y})$ satisfies $R(x, \mathbf{x}, M, s_1)$ and $\mathbf{x} \notin \mathbf{x}$.

For CCA2 both A_1 and A_2 get the decryption oracle, but A_2 is not allowed to call it on the challenge ciphertext y; for CCA1 just A_1 gets the decryption oracle; and for CPA neither A_1 nor A_2 get it. However, a key feature of the SNM definition is that no matter what the attack, the simulator does not get a decryption oracle, even though the adversary may get one.

We now provide a formal definition for simulation-based non-mall eability. When we say $\mathcal{O}_i = \varepsilon$, where $i \in \{1, 2\}$, we mean \mathcal{O}_i is the function which, on any input, returns the empty string, ε .

Definition 1. [SNM-CPA, SNM-CCA1, SNM-CCA2] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let R be a relation, let $A = (A_1, A_2)$ be an adversary, and let $S = (S_1, S_2)$ be an algorithm (the "simulator"). For atk $\in \{\text{cpa}, \text{cca1}, \text{cca2}\}$ and $k \in \mathbb{N}$ define

 $\mathsf{Adv}^{\mathsf{snm-atk}}_{A,S,\Pi}(R,k) \stackrel{\mathsf{def}}{=} \Pr[\mathsf{Expt}^{\mathsf{snm-atk}}_{A,\Pi}(R,k) = 1] - \Pr[\mathsf{Expt}^{\mathsf{snm-atk}}_{S,\Pi}(R,k) = 1] \ ,$ where

$$\begin{split} \mathsf{Expt}^{\mathsf{snm-atk}}_{A,\Pi}(R,k) & \mathsf{Expt}^{\mathsf{snm-atk}}_{S,\Pi}(R,k) \\ & (pk,sk) \leftarrow \mathcal{K}(1^k) \\ & (M,s_1,s_2) \leftarrow A_1^{\mathcal{O}_1}(pk) \\ & x \leftarrow M \\ & y \leftarrow \mathcal{E}_{pk}(x) \\ & \mathbf{y} \leftarrow A_2^{\mathcal{O}_2}(s_2,y) \\ & \mathbf{x} \leftarrow \mathcal{D}_{sk}(\mathbf{y}) \\ & \mathsf{return} \ 1 \ \mathrm{iff} \ (y \not\in \mathbf{y}) \land R(x,\mathbf{x},M,s_1) \end{split} \qquad \begin{aligned} & \mathsf{Expt}^{\mathsf{snm-atk}}_{S,\Pi}(R,k) \\ & (pk,sk) \leftarrow \mathcal{K}(1^k) \\ & (M,s_1,s_2) \leftarrow S_1(pk) \\ & x \leftarrow M \end{aligned}$$

and

If atk = cpa then
$$\mathcal{O}_1(\cdot) = \varepsilon$$
 and $\mathcal{O}_2(\cdot) = \varepsilon$
If atk = cca1 then $\mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot)$ and $\mathcal{O}_2(\cdot) = \varepsilon$
If atk = cca2 then $\mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot)$ and $\mathcal{O}_2(\cdot) = \mathcal{D}_{sk}(\cdot)$

We say that Π is secure in the sense of SNM-ATK if for every polynomial p(k), every R computable in time p(k), every A that runs in time p(k) and outputs a valid message space M sampleable in time p(k), there exists a polynomial-time algorithm $S = (S_1, S_2)$ such that $Adv_{A,S,\Pi}^{snm-atk}(R, k)$ is negligible. It is understood above that in the case of CCA2, A_2 is not allowed to ask its oracle for the decryption of the challenge ciphertext y.

The condition that $y \notin \mathbf{y}$ is made in order to not give the adversary credit for the trivial and unavoidable action of copying the challenge ciphertext. The requirement that M is valid is important; it stems from the fact that encryption is not intended to conceal the length of the plaintext.

3.2 Definition of CNM

We now recall the definition of non-malleable encryption of [2], which we call CNM for "comparison based non-malleability." Let $A = (A_1, A_2)$ be an adversary. The first stage of the adversary, namely A_1 , is given the public key pk, and outputs a description of a valid message space, described by a sampling algorithm M. The second stage of the adversary, namely A_2 , receives an encryption y of a random message x drawn from M. The adversary then outputs a (description of a) relation R and a vector \mathbf{y} . We insist that no component of \mathbf{y} be equal to y. The adversary hopes that $R(x, \mathbf{x})$ holds, where $\mathbf{x} \leftarrow \mathcal{D}_{sk}(\mathbf{y})$. An adversary (A_1, A_2) is successful if it can do this with a probability significantly more than that with which $R(x, \mathbf{x})$ holds if she had been given as input not an encryption of x but rather an encryption of some \tilde{x} also chosen uniformly from M, independently of x.

Definition 2. [CNM-CPA, CNM-CCA1, CNM-CCA2] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and let $A = (A_1, A_2)$ be an adversary. For atk $\in \{\text{cpa}, \text{cca1}, \text{cca2}\}$ and $k \in \mathbb{N}$ define

$$\mathsf{Adv}^{\mathrm{cnm-atk}}_{A,\Pi}(k) \ \stackrel{\mathrm{def}}{=} \ \Pr[\mathsf{Expt}^{\mathrm{cnm-atk}}_{A,\Pi}(k) = 1] - \Pr[\widetilde{\mathsf{Expt}}^{\mathrm{cnm-atk}}_{A,\Pi}(k) = 1] \ ,$$

where

$$\begin{array}{l} \operatorname{Expt}_{A,\Pi}^{\operatorname{cnm-atk}}(k) & \operatorname{\widetilde{Expt}}_{A,\Pi}^{\operatorname{cnm-atk}}(k) \\ (pk,sk) \leftarrow \mathcal{K}(1^k) & (pk,sk) \leftarrow \mathcal{K}(1^k) \\ (M,s) \leftarrow A_1^{\mathcal{O}_1}(pk) & (pk,sk) \leftarrow \mathcal{K}(1^k) \\ x \leftarrow M & y \leftarrow \mathcal{E}_{pk}(x) & (M,s) \leftarrow A_1^{\mathcal{O}_1}(pk) \\ (R,\mathbf{y}) \leftarrow A_2^{\mathcal{O}_2}(s,y) & \tilde{x} \leftarrow \mathcal{D}_{sk}(\tilde{\mathbf{y}}) \\ \mathbf{x} \leftarrow \mathcal{D}_{sk}(\mathbf{y}) & \operatorname{return 1 iff} (y \not\in \mathbf{y}) \wedge R(x,\mathbf{x}) & \operatorname{return 1 iff} (\tilde{y} \not\in \tilde{\mathbf{y}}) \wedge R(x,\tilde{\mathbf{x}}) \end{array}$$

and

```
If atk = cpa then \mathcal{O}_1(\cdot) = \varepsilon and \mathcal{O}_2(\cdot) = \varepsilon
If atk = cca1 then \mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot) and \mathcal{O}_2(\cdot) = \varepsilon
If atk = cca2 then \mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot) and \mathcal{O}_2(\cdot) = \mathcal{D}_{sk}(\cdot)
```

We say that Π is secure in the sense of CNM-ATK if for every polynomial p(k): if A runs in time p(k), outputs a (valid) message space M sampleable in time p(k), and outputs a relation R computable in time p(k), then $\mathsf{Adv}_{A,\Pi}^{\mathsf{cnm}-\mathsf{atk}}(\cdot)$ is negligible. It is understood above that in the case of CCA2, A_2 is not allowed to ask its oracle for the decryption of the challenge ciphertext y.

We declare the adversary unsuccessful when some ciphertext $\mathbf{y}[i]$ does not have a valid decryption (that is, $\bot \in \mathbf{x}$), because in this case, the receiver is simply going to reject the adversary's message anyway.

The major difference between SNM and CNM is that the former asks for a simulator and the latter does not, but some more minor differences exist too. For example in SNM the relation R is fixed beforehand, while in CNM it is generated dynamically by the adversary in its second stage.

4 New Notion: IND under Parallel Attack

We present a new notion of security for a public key encryption scheme: indistinguishability under a parallel chosen-ciphertext attack.

Here, mall eability is not evident in any explicit way; there is no relation R, and the adversary does not output ciphertexts, but rather tries to predict information about the plaintext. Nonetheless we show that this notion is equivalent to both forms of non-malleability given above.

In this attack, the adversary is allowed to query the decryption oracle a polynomial number of times, but the different queries made are not allowed to depend on each other. A simple way to visualize this is to imagine the adversary making the queries "in parallel," as a vector \mathbf{c} where $\mathbf{c}[1], \ldots, \mathbf{c}[n]$ are ciphertexts, for $n = |\mathbf{c}|$. The oracle replies with $\mathcal{D}_{sk}(\mathbf{c}) = (\mathcal{D}_{sk}(\mathbf{c}[1]), \ldots, \mathcal{D}_{sk}(\mathbf{c}[n]))$, the vector of the corresponding plaintexts. Only one of these parallel queries is allowed, and it is always in the second stage, meaning can be a function of the challenge ciphertext.

It is convenient to make the parallel query quite explicit in the formalization. We do this by considering the second stage A_2 of the adversary as made up of two sub-stages, $A_{2,q}$ and $A_{2,q}$, the "q" standing for "query" and the "g" for "guess". The first stage outputs the parallel query; the second is given the response, and completes the guessing work of the second stage. More precisely, view the adversary $A = (A_1, A_2)$ where $A_2 = (A_{2,q}, A_{2,q})$ as a pair. Algorithm A_1 is run on input the public key, pk. At the end of A_1 's execution she outputs a triple (x_0, x_1, s_1) , the first two components being messages which we insist be of the same length, and the last being state information (possibly including pk) which she wants to preserve. A random one of x_0 and x_1 is now selected, say x_b . A "challenge" y is determined by encrypting x_b under pk. It is the job of the pair $(A_{2,q}, A_{2,q})$ to try to determine if y was selected as the encryption of x_0 or x_1 , i.e. try to guess b. To make this determination, first run $A_{2,q}$ on x_0, x_1, s_1, y and let it output a parallel query c and state information s_2 . (The latter will include y, s_1 if necessary.) Then run $A_{2,q}$ on input $\mathcal{D}_{sk}(\mathbf{c}), s_2$ and get a guess g. The adversary wins if q = b.

We can add this parallel attack to any of the previous attacks CPA, CCA1, CCA2, yielding respectively the attacks PA0, PA1, PA2. Note that since in CCA2, the second stage of the adversary can already do adaptive chosen ciphertext attacks, giving it the ability to perform a parallel attack yields no additional power, so in fact CCA2 = PA2. For concision and clarity we simultaneously define indistinguishability with respect to PA0, PA1, and PA2. The only difference lies in whether or not A_1 and A_2 are given decryption oracles. We let the string atk be instantiated by any of the formal symbols pa0, pa1, pa2, while ATK is then the corresponding formal symbol from PA0, PA1, PA2.

Definition 3. [IND-PA0, IND-PA1, IND-PA2] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and let $A = (A_1, A_2)$ be an adversary. For atk $\in \{\text{pa0}, \text{pa1}, \text{pa2}\}$ and $k \in \mathbb{N}$, let

$$\mathsf{Adv}^{\mathrm{ind\text{-}atk}}_{A,\Pi}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{Expt}^{\mathrm{ind\text{-}atk}}_{A,\Pi}(k) = 1] - \frac{1}{2}$$

where

$$\begin{split} & \mathsf{Expt}^{\mathrm{ind-atk}}_{A,\Pi}(k) \\ & (pk,sk) \leftarrow \mathcal{K}(1^k) \\ & (x_0,x_1,s_1) \leftarrow A_1^{\mathcal{O}_1}(pk) \\ & b \leftarrow \{0,1\} \\ & y \leftarrow \mathcal{E}_{pk}(x_b) \\ & (\mathbf{c},s_2) \leftarrow A_{2,q}^{\mathcal{O}_2}(x_0,x_1,s_1,y) \\ & \mathbf{p} \leftarrow \mathcal{D}_{sk}(\mathbf{c}) \\ & g \leftarrow A_{2,g}^{\mathcal{O}_2}(\mathbf{p},s_2) \\ & \mathrm{return} \ 1 \ \mathrm{iff} \ (y \not\in \mathbf{c}) \land (g=b) \end{split}$$

and

```
If atk = pa0 then \mathcal{O}_1(\cdot) = \varepsilon and \mathcal{O}_2(\cdot) = \varepsilon
If atk = pa1 then \mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot) and \mathcal{O}_2(\cdot) = \varepsilon
If atk = pa2 then \mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot) and \mathcal{O}_2(\cdot) = \mathcal{D}_{sk}(\cdot)
```

We say that Π is secure in the sense of IND-ATK if A being polynomial-time implies that $\mathsf{Adv}_{A,\Pi}^{\mathsf{ind-atk}}(\cdot)$ is negligible. We insist, above, that A_1 outputs x_0, x_1 with $|x_0| = |x_1|$. In the case of PA2, we further insist that A_2 does not ask its oracle to decrypt y.

5 Results

Here we establish the equivalence of all three notions discussed. This is established by the following sequence of propositions.

Proposition 1. [CNM-ATK \Rightarrow SNM-ATK] For any ATK \in {CPA, CCA1, CCA2}, if encryption scheme Π is secure in the sense of CNM-ATK then Π is secure in the sense of SNM-ATK.

Proposition 2. [SNM-ATK \Rightarrow IND-PXX] For any ATK \in {CPA, CCA1, CCA2}, if encryption scheme Π is secure in the sense of SNM-ATK then Π is secure in the sense of IND-PXX, where

```
If ATK = CPA then PXX = PA0

If ATK = CCA1 then PXX = PA1

If ATK = CCA2 then PXX = PA2
```

Proposition 3. [IND-PXX \Rightarrow CNM-ATK] For any PXX \in {PA0, PA1, PA2}, if encryption scheme Π is secure in the sense of IND-PXX then Π is secure in the sense of CNM-ATK, where

```
If PXX = PA0 then ATK = CPA
If PXX = PA1 then ATK = CCA1
If PXX = PA2 then ATK = CCA2
```

We now turn to the proofs.

5.1 CNM Implies SNM

How does SNM compare with CNM? Let us first consider the question under CPA. It is easy to see that CNM-CPA \Rightarrow SNM-CPA. Intuitively, the CNM-CPA definition can be viewed as requiring that for every adversary A there exist a specific type of simulator, which we can call a "canonical simulator," $A' = (A'_1, A'_2)$. The first stage, A'_1 , is identical to A_1 . The second simulator stage A_2 simply chooses a random message from the message space M that was output by A'_1 , and runs the adversary's second stage A_2 on the encryption of that message. Since A does not have a decryption oracle, A' can indeed do this. With some additional appropriate tailoring we can construct a simulator that meets the conditions of the definition of SNM-CPA.

Let us try to extend this line of thought to CCA1 and CCA2. If we wish to continue to think in terms of the canonical simulator, the difficulty is that this "simulator" would, in running A, now need access to a decryption oracle, which is not allowed under SNM. Thus, it might appear that CNM-CPA is actually weaker, corresponding to the ability to simulate by simulators which are also given the decryption oracle.

However, this appearance is false. In fact, CNM-CPA is not weaker; rather, CNM-ATK implies SNM-ATK for all three types of attacks ATK, including CCA1 and CCA2. (In other words, if a scheme meets the CNM-CPA definition, it is possible to design a simulator according to the SNM definition.)

Proof of Proposition 1: Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the given encryption scheme. Let R and $A = (A_1, A_2)$ be given. To show the scheme is secure in the sense of SNM-ATK we need to construct a simulator $S = (S_1, S_2)$. The idea is that S will run A on a newly chosen public key of which it knows the corresponding decryption key:

$$\begin{array}{l} \operatorname{Algorithm} S_1(pk) \\ (pk',sk') \leftarrow \mathcal{K}(1^k) \\ (M,s_1,s_2) \leftarrow A_1^{\tilde{\mathcal{O}}_1}(pk') \\ \tilde{s}_2 \leftarrow (M,s_2,pk,pk',sk') \\ \operatorname{return} (M,s_1,\tilde{s}_2) \end{array} \begin{array}{l} \operatorname{Algorithm} S_2(\tilde{s}_2) \text{ where } \tilde{s}_2 = (M,s_2,pk,pk',sk') \\ \tilde{x} \leftarrow M \; ; \; \tilde{y} \leftarrow \mathcal{E}_{pk'}(\tilde{x}) \\ \tilde{y} \leftarrow A_2^{\tilde{\mathcal{O}}_2}(s_2,\tilde{y}) \\ \operatorname{if} \; (\tilde{y} \in \tilde{\mathbf{y}}) \; \operatorname{then \; abort} \\ \tilde{\mathbf{x}} \leftarrow \mathcal{D}_{sk'}(\tilde{\mathbf{y}}) \\ \mathbf{y} \leftarrow \mathcal{E}_{pk}(\tilde{\mathbf{x}}) \\ \operatorname{return} \mathbf{y} \end{array}$$

where

If atk = cpa then
$$\tilde{\mathcal{O}}_1(\cdot) = \varepsilon$$
 and $\tilde{\mathcal{O}}_2(\cdot) = \varepsilon$
If atk = cca1 then $\tilde{\mathcal{O}}_1(\cdot) = \mathcal{D}_{sk'}(\cdot)$ and $\tilde{\mathcal{O}}_2(\cdot) = \varepsilon$
If atk = cca2 then $\tilde{\mathcal{O}}_1(\cdot) = \mathcal{D}_{sk'}(\cdot)$ and $\tilde{\mathcal{O}}_2(\cdot) = \mathcal{D}'_{sk'}(\cdot)$

A key point is that the simulator, being in possession of sk', can indeed run A with the stated oracles. (That's how it avoids needing access to the "real" oracles $\mathcal{O}_1, \mathcal{O}_2$ that are provided to A and might depend on sk.) Now we want to show that $\mathsf{Adv}_{A,S,\Pi}^{\mathsf{snm-atk}}(R,k)$ is negligible. We will do this using the assumption that Π is secure in the sense of CNM-ATK. To that end, we consider the following adversary $B = (B_1, B_2)$ attacking Π in the sense of CNM-ATK.

$$\begin{array}{l} \operatorname{Algorithm} \left. B_1^{\mathcal{O}_1}(pk) \right. \\ \left. (M,s_1,s_2) \leftarrow A_1^{\mathcal{O}_1}(pk) \right. \\ \operatorname{return} \left. (M,(M,s_1,s_2)) \right| & \operatorname{Define} \left. R' \text{ by } R'(a,\mathbf{b}) = 1 \text{ iff } R(a,\mathbf{b},M,s_1) = 1 \\ \mathbf{y} \leftarrow A_2^{\mathcal{O}_2}(s_2,y) \\ \operatorname{return} \left. (R',\mathbf{y}) \right. \end{array}$$

It is clear from the definition of B that $\mathsf{Expt}_{B,\Pi}^{\mathsf{cnm-atk}}(k)$ is precisely the same as $\mathsf{Expt}_{A,\Pi}^{\mathsf{snm-atk}}(R,k)$. Now, let us expand the definition of $\mathsf{Expt}_{S,\Pi}^{\mathsf{snm-atk}}(R,k)$, substituting in the definition of S given above. Once we eliminate lines that do not affect the outcome of the experiment, this yields:

```
\begin{split} & \mathsf{Expt}^{\mathsf{snm-atk}}_{S,\Pi}(R,k) \\ & (pk',sk') \leftarrow \mathcal{K}(1^k) \\ & (M,s_1,s_2) \leftarrow A_1^{\tilde{\mathcal{O}}_1}(pk') \\ & x, \tilde{x} \leftarrow M \\ & \tilde{y} \leftarrow \mathcal{E}_{pk'}(\tilde{x}) \\ & \tilde{\mathbf{y}} \leftarrow A_2^{\tilde{\mathcal{O}}_2}(s_2,\tilde{y}) \\ & \tilde{\mathbf{x}} \leftarrow \mathcal{D}_{sk'}(\tilde{\mathbf{y}}) \\ & \text{return 1 iff } (\tilde{y} \notin \tilde{\mathbf{y}}) \land R(x,\tilde{\mathbf{x}},M,s_1) \end{split}
```

Thus, glancing at the definition of B, we see that this experiment is precisely the same as $\widetilde{\mathsf{Expt}}_{B,\Pi}^{\mathsf{cnm-atk}}(k)$ with pk and sk replaced by the pk' and sk' chosen by the simulator. Hence, $\mathsf{Adv}_{A,S,\Pi}^{\mathsf{snm-atk}}(R,k) = \mathsf{Adv}_{B,\Pi}^{\mathsf{cnm-atk}}(k)$. But the latter is negligible since Π is secure in the sense of CNM-ATK, so the former is negligible too.

5.2 SNM-ATK \Rightarrow IND-PXX

Proof of Proposition 2: We already know that SNM-CCA2 and IND-CCA2 are equivalent [6]. But IND-PA2 and IND-CCA2 are obviously identical since in both cases, a chosen-ciphertext attack is allowed in the second stage, and this subsumes a parallel attack. Thus we need prove the proposition only for the cases of ATK = CPA and ATK = CCA1.

We are assuming that encryption scheme Π is secure in the SNM-ATK sense. We will show it is also secure in the IND-PXX sense. Let $B=(B_1,B_2)$ be an IND-PXX adversary attacking Π . We want to show that $\operatorname{Adv}_{B,\Pi}^{\operatorname{ind-pxx}}(\cdot)$ is negligible. To this end, we describe a relation R and an SNM-ATK adversary $A=(A_1,A_2)$ attacking Π using R. We wish to show that A will have the same advantage attacking Π using R as B has as an IND-PXX adversary using a parallel attack. What allows us to do this is to pick the relation R to capture the success condition of B's parallel attack. Adversaries A and B have access to an oracle \mathcal{O}_1 in their first stage (but we can assume that oracle in their second stage $\mathcal{O}_2=\varepsilon$), with this oracle being instantiated according to the attack ATK as per the definitions.

To get some intuition it is best to think of ATK = CPA, meaning A is allowed only a chosen-plaintext attack. However, B has (limited) access to a decryption oracle; it is allowed the parallel query. How then can A "simulate" B? The key observation is that the non-malleability goal involves an "implicit" ciphertext attack on the part of the adversary, even under CPA. This arises from the ciphertext vector \mathbf{y} that such an adversary outputs. It gets decrypted, and the results are fed into the relation R. Thus, the idea of our proof is to make A output, as its final response, the parallel query that B will make. Now, B would expect to get back the response and compute with it, which A can't do; once it has output its final ciphertext, it stops, and the relation R gets evaluated on the corresponding plaintext. So we define R in such a way that it "completes" B's computation. A useful way to think about this is as if A were trying to "communicate" with R, passing it the information that R needs to execute B.

Notice that for this to work it is crucial that B's query is a parallel one. If B were making the usual adaptive queries, A could not output a single ciphertext vector, because it would have to know the decryption of the first ciphertext query before it would even know the ciphertext which is the second query. Yet, for the non-malleability game, A must output a single vector.

This is the rough idea. There are a couple of subtleties. R needs to know several pieces of information that depend on the computation of some stages of B, such as coin tosses. A must communicate them to R. The only mechanism that A has to communicate with R is via the ciphertext vector \mathbf{y} that A outputs, whose decryption is fed to R. So any information that A wants to pass along, it encrypts and puts in this vector.

Now let us provide a more complete description. Given the IND-PXX adversary $B = (B_1, B_{2,q}, B_{2,g})$, we want to define the SNM-ATK adversary $A = (A_1, A_2)$. The first stage A_1 is:

```
Algorithm A_1^{\mathcal{O}_1}(pk)

(m_0, m_1, t_1) \leftarrow B_1^{\mathcal{O}_1}(pk)

Let M be a canonical encoding of the uniform distribution over \{m_0, m_1\}

along with an encoding of the ordered pair (m_0, m_1)

Let s_1 = \varepsilon and s_2 = (m_0, m_1, t_1, pk)

return (M, s_1, s_2)
```

The second stage A_2 is:

```
Algorithm A_2(s_2) where s_2 = (m_0, m_1, t_1, pk)

(\mathbf{c}, t_2) \leftarrow B_{2,q}(m_0, m_1, t_1, y)

Choose random coins \sigma for B_{2,g}

e_1 \leftarrow \mathcal{E}_{pk}(t_2) \; ; \; e_2 \leftarrow \mathcal{E}_{pk}(\sigma)

Let \mathbf{y} = (e_1, e_2, \mathbf{c}[1], \dots, \mathbf{c}[|\mathbf{c}|])

return \mathbf{y}
```

Notice above that A_2 picks coins σ for $B_{2,g}$. We can think of each stage of B as picking its own coins afresh, since any information needing to be communicated from stage to stage is passed along in the state information. Now, here is the relation R.

```
Relation R(x, \mathbf{x}, M, s_1)

Let t_2 and \sigma be the first two components of \mathbf{x}

Let the remaining components form the vector \mathbf{p}

If (M is not a valid canonical encoding of an ordered pair of strings (m_0, m_1)

and the uniform distribution over \{m_0, m_1\})

then return 0

Let b be such that x = m_b

return 1 iff B_{2,g}(\mathbf{p}, t_2; \sigma) = b
```

Notice that R is polynomial time computable.

If one expands the definition of $\mathsf{Expt}_{A,\Pi}^{\mathsf{snm-atk}}(R,k)$ using the definitions of R and A above, by eliminating unnecessary lines we see that the experiment is the same as $\mathsf{Expt}_{B,\Pi}^{\mathsf{ind-atk}}(k)$ up to negligible factors.

To conclude the proof, we need only show that the probability that any simulator S will succeed in attacking R as defined above in the experiment $\mathsf{Expt}_{S,\Pi}^{\mathsf{snm-atk}}(R,k)$ is at most $\frac{1}{2}$. By construction, in order to satisfy R the first stage of S must output a distribution M that is uniform on two messages m_0 and m_1 . Suppose S does so with probability $q \leq 1$. Now let p_b be the probability that S create output so as to cause $B_{2,g}$ to output b. Since the behavior of S is

independent of the chosen $x \in M$, $p_0 + p_1 \le 1$. Hence, also by independence, the probability that S will succeed in causing R to accept is bounded by $q(p_0 \cdot \Pr[x = m_0] + p_1 \cdot \Pr[x = m_1]) \le q(p_0 \cdot \frac{1}{2} + p_1 \cdot \frac{1}{2}) \le \frac{1}{2}$.

Thus, we have that

$$\begin{split} \mathsf{Adv}^{\mathrm{ind-atk}}_{B,\Pi}(k) &= \Pr[\mathsf{Expt}^{\mathrm{ind-atk}}_{B,\Pi}(k) = 1] - \frac{1}{2} \\ &\leq \Pr[\mathsf{Expt}^{\mathrm{snm-atk}}_{A,\Pi}(R,k) = 1] - \frac{1}{2} + \epsilon(k) \\ &\leq \Pr[\mathsf{Expt}^{\mathrm{snm-atk}}_{A,\Pi}(R,k) = 1] - \Pr[\mathsf{Expt}^{\mathrm{snm-atk}}_{S,\Pi}(R,k) = 1] + \epsilon(k) \\ &\leq \mathsf{Adv}^{\mathrm{snm-atk}}_{A,\Pi}(R,k) + \epsilon(k) \end{split}$$

where $\epsilon(k)$ is a negligible function. Since by assumption $\mathsf{Adv}_{A,\Pi}^{\mathsf{snm-atk}}(R,k)$ is negligible, this completes the proof.

5.3 IND-PXX \Rightarrow CNM-ATK

Proof of Proposition 3: We are assuming that Π is secure in the IND-PXX sense. We will show it is also secure in the CNM-ATK sense.

Let $B = (B_1, B_2)$ be an CNM-ATK adversary attacking Π . We want to show that $\mathsf{Adv}_{B,\Pi}^{\mathsf{cnm-atk}}(\cdot)$ is negligible. To this end, we describe an IND-PXX adversary $A = (A_1, A_2)$ attacking Π . We wish to show that A will have the same advantage as an IND-PXX adversary as B has as an CNM-ATK adversary. The definition of a parallel attack was chosen to make this proof easy, and the intuition will be simple: since A has access to a parallel decryption oracle in the second stage, she can decrypt the ciphertexts that B outputs, and check herself to see if B's relation holds.

Given the CNM-ATK adversary $B = (B_1, B_2)$, we define the IND-PXX adversary $A = (A_1, A_{2,q}, A_{2,q})$ as follows:

$$\begin{aligned} & \text{Algorithm } A_1^{\mathcal{O}_1}(pk) \\ & (M,t) \leftarrow B_1^{\mathcal{O}_1}(pk) \\ & x_0, x_1 \leftarrow M \ ; \ s_1 \leftarrow (M,t) \\ & \text{return } (x_0, x_1, s_1) \end{aligned}$$

Algorithm
$$A_{2,q}^{\mathcal{O}_2}(x_0,x_1,s_1,y)$$
 where $s_1=(M,t)$ $(R,\mathbf{c}) \leftarrow B_2^{\mathcal{O}_2}(M,t,y)$ $s_2 \leftarrow (R,x_0,x_1,\mathbf{c},y)$ return (\mathbf{c},s_2)

Algorithm
$$A_{2,q}^{\mathcal{O}_2}(\mathbf{p},s_2)$$
 where $s_2=(R,x_0,x_1,\mathbf{c},y)$

```
\begin{array}{c} \text{if } (y \notin \mathbf{c}) \land R(x_0, \mathbf{p}) \\ \text{then } g \leftarrow 0 \\ \text{else } g \leftarrow \{0, 1\} \\ \text{return } g \end{array}
```

A straightforward calculation establishes that the advantage of the IND-PXX adversary given above will be negligibly close to the advantage of the CNM-ATK adversary, completing the proof.

Acknowledgments

Mihir Bellare was supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering. Amit Sahai was supported by a DOD/NDSEG Graduate Fellowship and partially by DARPA grant DABT-96-C-0018.

References

- M. Bellare, R. Canetti and H. Krawczyk, A modular approach to the design and analysis of authentication and key exchange protocols. *Proceedings of the* 30th Annual Symposium on Theory of Computing, ACM, 1998.
- M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, Relations among notions of security for public-key encryption schemes. Advances in Cryptology - Crypto 98 Proceedings, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
- 3. M. Bellare and P. Rogaway, Optimal asymmetric encryption How to encrypt with RSA. Advances in Cryptology Eurocrypt 94 Proceedings, Lecture Notes in Computer Science Vol. 950, A. De Santis ed., Springer-Verlag, 1994.
- R. CRAMER AND V. SHOUP, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. Advances in Cryptology – Crypto 98 Proceedings, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
- D. Dolev, C. Dwork, and M. Naor, Non-malleable cryptography. Proceedings of the 23rd Annual Symposium on Theory of Computing, ACM, 1991. Also Technical Report CS95-27, Weizmann Institute of Science, 1995.
- D. Dolev, C. Dwork, and M. Naor, Non-malleable cryptography. Manuscript, 1998.
- O. Goldreich, A uniform complexity treatment of encryption and zeroknowledge. *Journal of Cryptology*, Vol. 6, 1993, pp. 21-53.
- S. GOLDWASSER AND S. MICALI, Probabilistic encryption. Journal of Computer and System Sciences, 28:270–299, 1984.
- 9. S. Halevi and H. Krawczyk, Public-key cryptography and password protocols. Proceedings of the Fifth Annual Conference on Computer and Communications Security, ACM, 1998.
- 10. S. MICALI, C. RACKOFF AND R. SLOAN, The notion of security for probabilistic cryptosystems. SIAM J. of Computing, April 1988.

- M. NAOR AND M. YUNG, Public-key cryptosystems provably secure against chosen ciphertext attacks. Proceedings of the 22nd Annual Symposium on Theory of Computing, ACM, 1990.
- 12. C. RACKOFF AND D. SIMON, Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *Advances in Cryptology Crypto* 91 *Proceedings*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.
- 13. SETCo (Secure Electronic Transaction LLC), The SET standard book 3 formal protocol definitions (version 1.0). May 31, 1997. Available from http://www.setco.org/
- 14. A. Yao, Theory and applications of trapdoor functions. Proceedings of the 23rd Symposium on Foundations of Computer Science, IEEE, 1982.

Secure Integration of Asymmetric and Symmetric Encryption Schemes

Eiichiro Fujisaki and Tatsuaki Okamoto

NTT Laboratories
1-1 Hikarinooka, Yokosuka-shi, 239-0847 JAPAN {fujisaki,okamoto}@sucaba.isl.ntt.co.jp

Abstract. This paper shows a generic and simple conversion from weak asymmetric and symmetric encryption schemes into an asymmetric encryption scheme which is secure in a very strong sense — indistinguishability against adaptive chosen-ciphertext attacks in the random oracle model. In particular, this conversion can be applied efficiently to an asymmetric encryption scheme that provides a large enough coin space and, for every message, many enough variants of the encryption, like the ElGamal encryption scheme.

Key words: Indistinguishability, Adaptive chosen-ciphertext attack, Random oracle model, Hybrid encryption.

1 Introduction

Suppose that an asymmetric encryption scheme is secure in a very weak sense — an adversary can't entirely decrypt the encryption of a random plaintext. Suppose that a symmetric encryption scheme is secure in the following weak sense — for all possible messages, m_1 and m_2 , in the indicated message space, an adversary can't distinguish the encryption of m_1 from the encryption of m_2 (where the adversary is not given the ability to encrypt or decrypt desired strings). From these schemes, we construct a new asymmetric encryption scheme. The (hybrid) encryption of a plaintext m is

$$\mathcal{E}_{pk}^{\mathrm{hy}}\left(m\right) = \hspace{0.2cm} \mathcal{E}_{pk}^{\mathrm{asym}}\left(\sigma; H(\sigma,m)\right) \hspace{0.2cm} \mid\mid \hspace{0.2cm} \mathcal{E}_{G(\sigma)}^{\mathrm{sym}}(m),$$

where

- σ is a random string chosen from an appropriate domain,
- $\mathcal{E}_{pk}^{\text{asym}}$ (message; coins) indicates the asymmetric encryption of the indicated message using the indicated coins as random bits,
- $\mathcal{E}_a^{\text{sym}}$ (messgage) indicates the symmetric encryption of the indicated message using the indicated string a, and
- G and H denote hash functions.

In the random oracle model (namely, G and H are modeled as random oracles), this hybrid encryption scheme is secure in a very strong sense — indistinguishability against adaptive chosen-ciphertext attacks.

We will provide the concrete security reduction in the exact security manner [3]. The security of this hybrid encryption scheme depends only on those of asymmetric and symmetric encryption primitives and the following property of the asymmetric encryption primitive— given an appropriate message space, for any message in the space, the variants of the encryption occur in a large enough number, provided the coins are chosen uniformly from the coin space of the encryption scheme. We will define the exact definition later and will also show, for any encryption scheme, a slightly modified scheme can provide an enough number of the variants. In particular, this conversion can be efficiently applied to an asymmetric encryption scheme with a large coin space, like the ElGamal encryption scheme.

1.1 Related Works

To create a practical and provably secure encryption scheme is one of important goals in cryptography. Although theoretical works have been done in many literatures [18, 13, 14, 19, 10], there are not so many schemes that satisfy both provable security and efficiency. In this section, we will refer to several schemes that are practical and provably secure in a very strong sense, such as [23, 3, 8, 1, 12], and will discuss these schemes (including ours) in the following terms.

Conversion A promising way to construct a practical and provably-secure encryption scheme is to convert it from primitives which are secure in a weaker sense.

In CRYPTO'94, Bellare and Rogaway presented a generic and simple conversion from a one-way trapdoor permutation (OWTP) such as the RSA primitive into an asymmetric encryption scheme which is secure in a very strong sense in the random oracle model [3]. A scheme created in this way is called OAEP (Optimal Asymmetric Encryption Padding). The strong security notion is indistinguishability against adaptive chosen-ciphertext attacks (IND-CCA), as described in [19]. However, the method in [3] was not applied to asymmetric encryption schemes. Therefore, several (practical) asymmetric encryption schemes lie outside the range of OAEP conversion, e.g., the ElGamal, Blum-Goldwasser, and Okamoto-Uchiyama encryption schemes [11, 6, 17].

Before their proposal, Zheng and Seberry had also proposed some practical schemes [23] aiming at chosen-cipher security, and, according to [3], in the random oracle model at least one of their schemes enjoys the same security as OAEP. That scheme too is, however, applied only to OWTPs.

The current authors recently presented a generic conversion from an asymmetric encryption scheme into an asymmetric one that is secure in the IND-CCA sense in the random oracle model [12]. However the security requirement of the primitive encryption scheme is stronger than that of [3] — the OAEP conversion starts from a OWTP while the conversion in [12] does from an asymmetric

encryption scheme which is secure in the sense of indistinguishability against chosen-plaintext attacks (IND-CPA).

Other conversion have been reported, such as [1,22,21]. However their schemes depend strongly on the primitive encryptions, they don't work for generic methods.

To the best of our knowledge, there has up to now not been proposed any generic (and efficient) method to convert an asymmetric encryption scheme into an IND-CCA secure one.

Hybrid Encryption An asymmetric encryption scheme is usually employed only for distributing a secret-key of a symmetric encryption scheme for message encryption. Actually, the hybrid usage of asymmetric and symmetric encryption schemes is very common in practice. On the other hand, hybrid usage is insecure in general, even if both the asymmetric and symmetric encryption schemes are secure in very strong senses. In spite of the fact that hybrid usage is common and that, in general, this is insecure, there has been little research on this subject; see [1, 8].

In [1], Abdalla, Bellare, and Rogaway present a hybrid encryption scheme, called DHAES, and prove that hybrid usage is secure in the IND-CCA sense in the random oracle model (or a strong assumption in the standard (not random oracle) model). The main difference from our work is that they use one more cryptographic primitive — message authentication code (MAC). In addition, their scheme depends on the Diffie-Hellman key-distribution scheme. DHAES is composed of the Diffie-Hellman key-distribution, a hash function, a symmetric encryption, and a message authentication code (MAC).

Cramer and Shoup briefly mentioned in their work that their scheme can be applied to hybrid usage with a symmetric encryption scheme [8].

1.2 Our Results

The contributions of this paper are twofold: One is to show a *generic* conversion from a very weak asymmetric encryption to an asymmetric encryption scheme which is secure in a very strong sense (IND-CCA in the random oracle model). The other is to exhibit a *generic* hybrid conversion of asymmetric and symmetric encryption schemes, proving the security explicitly. Our conversion starts from arbitrary encryption schemes and each scheme so obtained is approximately as efficient as, or more efficient than, the previously proposed schemes [1, 8, 22, 21].

2 Preliminary

We begin with some notations.

Definition 1. Let A be a probabilistic algorithm and let $A(x_1, \ldots, x_n; r)$ be the result of running A on input (x_1, \ldots, x_n) and random coins r. We denote by $y \leftarrow A(x_1, \ldots, x_n)$ the experiment of picking r at random and letting y be

 $A(x_1, \ldots, x_n; r)$ (i.e., $y = A(x_1, \ldots, x_n; r)$). If S is a finite set, let $y \leftarrow_R S$ be the operation of picking y at random and uniformly from S. When S, T, \ldots , denote probability spaces, $\Pr[x \leftarrow S; y \leftarrow T; \cdots : p(x, y, \ldots)]$ denotes the probability that the predicate, $p(x, y, \ldots)$, is true after the experiments, $x \leftarrow S, y \leftarrow T, \cdots$, are executed in that order. Moreover, |x| denotes the bit length of string x and #S denotes the cardinality of set S.

Here we define asymmetric and symmetric encryption schemes, basically following [13, 3].

Definition 2. [Asymmetric Encryption] An asymmetric encryption scheme, $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathtt{COINS}, \mathtt{MSPC})$, is a triple of algorithm, associated with finite sets, $\mathtt{COINS}(k)$ and $\mathtt{MSPC}(k), \subseteq \{0,1\}^*$, for $k \in \mathbb{N}$, where

- K, called the key-generation algorithm, is a probabilistic algorithm which on input 1^k $(k \in \mathbb{N})$ outputs a pair of strings, $(pk, sk) \leftarrow K(1^k)$.
- \mathcal{E} , called the encryption algorithm, is a probabilistic algorithm that takes a pair of strings, pk and x, and a string $r \leftarrow \mathtt{COINS}(k)$, and produces a string $y = \mathcal{E}_{pk}(x;r)$.
- \mathcal{D} , called the decryption algorithm, is a deterministic algorithm that takes a pair of strings, sk and y, and returns a string $x \leftarrow \mathcal{D}_{sk}(y)$.

We require that, for any $k \in \mathbb{N}$, if $(pk, sk) \leftarrow \mathcal{K}(1^k)$, $x \in MSPC$, and $y \leftarrow \mathcal{E}_{pk}(x)$, then $\mathcal{D}_{sk}(y) = x$.

Definition 3. [Symmetric Encryption] A symmetric encryption scheme, $\Pi = (\mathcal{E}, \mathcal{D}, \text{KSPC}, \text{MSPC})$, is a pair of algorithms associated with finite sets, KSPC(k) and MSPC(k), $\subseteq \{0,1\}^*$, for $k \in \mathbb{N}$, where

- \mathcal{E} , called the encryption algorithm, is a deterministic algorithm that takes a pair of strings, a and x, and produces $y = \mathcal{E}_a(x)$.
- \mathcal{D} , called the decryption algorithm, is a deterministic algorithm that takes a pair of strings, a and y, and outputs a string $x = \mathcal{D}_a(y)$.

We require that, for any $k \in \mathbb{N}$, if $a \in KSPC(k)$, $x \in MSPC$, and $y = \mathcal{E}_a(x)$, then $\mathcal{D}_a(y) = x$.

3 Basic Conversion

In this section, we present our conversion.

Let $\Pi^{\operatorname{asym}} = (\mathcal{K}^{\operatorname{asym}}, \mathcal{E}^{\operatorname{asym}}, \mathcal{D}^{\operatorname{asym}}, \operatorname{COINS}^{\operatorname{asym}}, \operatorname{MSPC}^{\operatorname{asym}})$ be an asymmetric encryption scheme and let $\Pi^{\operatorname{sym}} = (\mathcal{E}^{\operatorname{sym}}, \mathcal{D}^{\operatorname{sym}}, \operatorname{KSPC}^{\operatorname{sym}}, \operatorname{MSPC}^{\operatorname{sym}})$ be a symmetric encryption scheme. Let $G: \operatorname{MSPC}^{\operatorname{asym}} \to \operatorname{KSPC}^{\operatorname{sym}}$ and $H: \operatorname{MSPC}^{\operatorname{asym}} \times \operatorname{MSPC}^{\operatorname{sym}} \to \operatorname{COINS}^{\operatorname{asym}}$ be hash functions.

From these primitives, we present a new asymmetric encryption scheme, $\Pi^{hy} = (\mathcal{K}^{hy}, \mathcal{E}^{hy}, \mathcal{D}^{hy}, \mathtt{COINS}^{hy}, \mathtt{MSPC}^{hy})$, (where $\mathtt{COIN}^{hy} = \mathtt{MSPC}^{asym}$ and $\mathtt{MSPC}^{hy} = \mathtt{MSPC}^{sym}$) as follows:

- Encryption

$$\mathcal{E}_{pk}^{\mathrm{hy}}(m;\sigma) = \mathcal{E}_{pk}^{\mathrm{asym}}(\sigma;H(\sigma,m)) \ || \ \mathcal{E}_{G(\sigma)}^{\mathrm{sym}}(m).$$

- Decryption

$$\mathcal{D}_{sk}^{\text{hy}}(c_1||c_2) = \begin{cases} \mathcal{D}_{G(\hat{\sigma})}^{\text{sym}}(c_2) & \text{if } c_1 = \mathcal{E}_{pk}^{\text{asym}}(\hat{\sigma}; H(\hat{\sigma}, \hat{m})), \\ \bot & \text{otherwise.} \end{cases}$$

where $\hat{\sigma} := \mathcal{D}_{sk}^{\operatorname{asym}}(c_1)$ and $\hat{m} := \mathcal{D}_{G(\hat{\sigma})}^{\operatorname{sym}}(c_2)$. (If there isn't $\mathcal{D}_{sk}^{\operatorname{asym}}(c_1)$ or $\mathcal{D}_{G(\hat{\sigma})}^{\operatorname{sym}}(c_2)$, then $\mathcal{D}_{sk}^{\operatorname{hy}}(c_1||c_2) = \perp$.)

Key-Generation $\mathcal{K}^{\mathrm{hy}}(1^k)$	Encryption $\mathcal{E}_{pk}^{\text{hy}}(m)$	Decryption $\mathcal{D}_{sk}^{ ext{hy}}(c_1,c_2)$
$(pk, sk) \leftarrow \mathcal{K}^{\operatorname{asym}}(1^k).$	$\begin{split} & \sigma \leftarrow_R \texttt{MSPC}^{\text{asym}}. \\ & r_1 := H(\sigma, m). \\ & r_2 := G(\sigma). \\ & c_1 := \mathcal{E}_{pk}^{\text{asym}}(\sigma; r_1). \\ & c_2 \leftarrow & \mathcal{E}_{r_2}^{\text{sym}}(m). \end{split}$	$\hat{\sigma} := \mathcal{D}_{sk}^{\mathrm{asym}}(c_1).$ $\hat{r_2} := G(\hat{\sigma}).$ $\hat{m} := \mathcal{D}_{r_2}^{\mathrm{sym}}(c_2).$ $\hat{r_1} := H(\hat{\sigma}, \hat{m}).$ If $c_1 == \mathcal{E}_{pk}^{\mathrm{asym}}(\hat{\sigma}; \hat{r_1})$
$\operatorname{return}\ (pk,sk).$	$\operatorname{return}\ (c_1,c_2).$	then $m := \hat{m}$, else $m := \perp$. return m .

Fig. 1. Hybrid encryption scheme

4 Security Definitions

4.1 Asymmetric Encryption

In this section we define security notions for asymmetric encryption.

One-way Encryption In the following, we give a very weak security notion (one-wayness) for an asymmetric encryption. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathtt{COINS},\mathtt{MSPC})$ be an asymmetric encryption. For Π , we consider an algorithm, A, called an adversary, that, taking a public-key, pk, outputted by \mathcal{K} , and an encryption, y, of a random plaintext in MSPC tries to decrypt y. The probability of A's success, denoted by the advantage of A, depends on A, Π , and the random choice of a plaintext from MSPC. A doesn't have any decryption oracle (while an encryption oracle doesn't matter because chosen-plaintext attacks are clearly unavoidable in an asymmetric encryption scheme).

Definition 4. [OWE] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathtt{COINS}, \mathtt{MSPC})$ be an asymmetric encryption scheme. Let A be an adversary. For $k \in \mathbb{N}$, define the advantage of A by $\mathrm{Adv}^{\mathrm{owe}}_{A,\Pi,\mathtt{MSPC}}(k) =$

$$\Pr[(pk, sk) \leftarrow \mathcal{K}(1^k); x \leftarrow \texttt{MSPC}(k); y \leftarrow \mathcal{E}_{pk}(x) : A(pk, y) = \mathcal{D}_{sk}(y)].$$

We say that adversary $A(t,\epsilon)$ -breaks Π in the sense of OWE if A runs in at most time t and achieves $\operatorname{Adv}_{A,\Pi}^{\operatorname{owe}}(k) \geq \epsilon$. We say that Π is (t,ϵ) -secure in the sense of OWE if there is no adversary that (t,ϵ) -breaks Π in that sense.

 γ -uniformity We introduce a property of asymmetric encryption in the following definition.

Definition 5. [γ -uniformity] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \text{COINS}, \text{MSPC})$ be an asymmetric encryption scheme. For given $(pk, sk) \in K(1^k)$, $x \in \text{MSPC}$ and $y \in \{0, 1\}^*$, define

$$\gamma(x,y) = \Pr[h \leftarrow_R \mathtt{COINS} : y = \mathcal{E}_{pk}(x;h)].$$

We say that Π is γ -uniform (for $k \in \mathbb{N}$), if, for any $(pk, sk) \in K(1^k)$, any $x \in \text{MSPC}$ and any $y \in \{0, 1\}^*$, $\gamma(x, y) \leq \gamma$.

Example 6. Let k be a security parameter. Define by ((y, g, p, q), x) a pair of public-key and secret-key where $y = g^x \mod p$, $x \in \mathbb{Z}/q\mathbb{Z}$, q = # < g > and k = |q|. The ElGamal encryption scheme, associated with 1^k , is then 2^{-k} -uniform.

Strong Security Notions We recall a classical and stronger security notion for an asymmetric encryption, called *indistinguishability* (IND), following [13, 4].

In this security notion, we consider an adversary, A, that takes two stages, find and guess. In the find stage, A takes public-key pk and returns two distinct messages, x_0, x_1 , and a string, s, to use in the next mode, and then, in the guess stage, takes the encryption of x_b , where $b \leftarrow_R \{0, 1\}$, and the above information, and tries to guess b. The advantage of A is meant by how well she can guess the value b. If A has the decryption oracle, $\mathcal{D}_{sk}(\cdot)$, we say that this experiment is an adaptive chosen-ciphertext attack (CCA), while, if A doesn't have it, we call it a chosen-plaintext attack (CPA).

The random oracle version of this security notion is defined by allowing A to make access to a random oracle (or plural oracles), which depends on Π . We define by Ω the map family from an appropriate domain to an appropriate range. The domain and range depend on the underlying encryption scheme, Π . Even if we choose two random functions that have distinct domains and distinct ranges respectively, we just write the experiment, for convenience, as $G, H \leftarrow \Omega$, instead of preparing two map families.

In the following definition, we define simultaneously indistinguishability with regard to CCA and CPA in the random oracle model.

Definition 7. [Indistinguishability] Let $\Pi = (K, \mathcal{E}, \mathcal{D}, \text{COINS}, \text{MSPC})$ be an asymmetric encryption scheme and let A be an Adversary. For $k \in \mathbb{N}$, define the following two advantages:

$$\begin{split} - \operatorname{Adv}^{\operatorname{ind-cpa}}_{A,H}(k) &= \\ 2 \cdot \Pr[G, H \leftarrow \varOmega; (pk, sk) \leftarrow \mathcal{K}(1^k); (x_0, x_1, s) \leftarrow A^{G,H}(\operatorname{find}, pk); \\ b \leftarrow_R \{0, 1\}; y \leftarrow \mathcal{E}^{G,H}_{nk}(x_b) : A^{G,H}(\operatorname{guess}, s, y) = b] - 1 \end{split}$$

$$- \operatorname{Adv}_{A,\Pi}^{\operatorname{ind-cca}}(k) =$$

$$\begin{aligned} 2 \cdot \Pr[G, H \leftarrow \varOmega; (pk, sk) \leftarrow \mathcal{K}(1^k); (x_0, x_1, s) \leftarrow A^{G, H, \mathcal{D}_{sk}}(\text{find}, pk); \\ b \leftarrow_R \{0, 1\}; y \leftarrow \mathcal{E}_{pk}^{G, H}(x_b) : A^{G, H, \mathcal{D}_{sk}}(\text{guess}, s, y) = b] - 1. \end{aligned}$$

We require that, for (x_0, x_1) that A outputs, $x_0 \neq x_1$ and $x_0, x_1 \in MSPC$.

We say that adversary A (t, q_a, q_h, ϵ) -breaks Π in the sense of IND-CPA in the random oracle model if A runs in at most time t, asks at most q_g queries to $G(\cdot)$, asks at most q_h queries to $H(\cdot)$, and achieves $\operatorname{Adv}_{A,\Pi}^{\operatorname{ind-cpa}}(k) \geq \epsilon$.

Similarly, we say that adversary A $(t, q_g, q_h, q_d, \epsilon)$ -breaks Π in the sense of IND-CCA in the random oracle model if A runs in at most time t, asks at most q_q queries to $G(\cdot)$, asks at most q_h queries to $H(\cdot)$, asks at most q_d queries to $\mathcal{D}_{sk}(\cdot)$, and achieves $\operatorname{Adv}_{A,\Pi}^{\operatorname{ind-cca}}(k) \geq \epsilon$.

We say that Π is (t, q_q, q_h, ϵ) -secure (or $(t, q_q, q_h, q_d, \epsilon)$ -secure) in the sense of IND-CPA (or IND-CCA) if there is no adversary that (t,q_g,q_h,ϵ) -breaks (or $(t, q_q, q_h, q_d, \epsilon)$ -breaks) Π in the corresponding sense.

Knowledge Extractor The notion of knowledge extractor for an asymmetric encryption scheme is defined in [3, 4]. We recall the definition, following [4]. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathtt{COINS}, \mathtt{MSPC})$ be an asymmetric encryption scheme. Let B and K be algorithms, called an adversary and a knowledge extractor, respectively. They work in the random oracle model as follows:

- Adversary B takes public-key pk and asks two kinds of queries, queries for random oracles, G and H, and queries for an encryption oracle, $\mathcal{E}_{pk}^{G,H}$, and, after taking the answers from those oracles, finally outputs a string y, where
 - \mathcal{T}_G denotes the set of all pairs of B's queries and the corresponding answers from G,
 - \mathcal{T}_H denotes the set of all pairs of B's queries and the corresponding answers from H,
 - \mathcal{Y} denotes the set of all answers recieved as ciphertexts from $\mathcal{E}_{nk}^{G,H}(\cdot)$.
 - y (output of B) is not in \mathcal{Y} .

We write the experiment above as $(\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, y) \leftarrow B^{G,H,\mathcal{E}_{pk}}(pk)$.

Here we insist that neither any query of B's to \mathcal{E}_{pk} is in \mathcal{Y} , nor any query of $\mathcal{E}_{pk}^{G,H}$'s to random oracles, G and H, is in \mathcal{T}_G and \mathcal{T}_H .

– Knowledge extractor K takes $(\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, y, pk)$ and outputs a string x.

Definition 8. [Knowledge Extractor] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathtt{COINS}, \mathtt{MSPC})$ be an asymmetric encryption scheme, let B be an adversary, and let K be a knowledge extractor. Define the following advantage: For $k \in \mathbb{N}$, let $\operatorname{Succ}_{KB,\Pi}^{\operatorname{ke}}(k) =$

$$\Pr[G, H \leftarrow \Omega; (pk, sk) \leftarrow \mathcal{K}(1^k); (\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, y) \leftarrow B^{G, H, \mathcal{E}_{pk}}(pk) : \\ K(\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, y, pk) = \mathcal{D}_{sk}(y)].$$

We say that B is a (q_g, q_h, q_e) -adversary if B takes pk, makes at most q_g queries to G, at most q_h queries to H and at most q_e queries to \mathcal{E}_{pk} respectively, and finally produces a string y. We say that K is a (t,λ) -knowledge extractor for B if K takes $(\mathcal{T},\mathcal{Y},y)$, runs in at most time t, and achieves $\operatorname{Succ}_{K,B,\Pi}^{\operatorname{ke}}(k) \geq \lambda$.

4.2 Symmetric Encryption

We prepare a security notion for symmetric encryptions, called find-guess. This notion is the symmetric encryption version of indistinguishability, following [5].

Let $\Pi = (\mathcal{E}, \mathcal{D}, \texttt{KSPC}, \texttt{MSPC})$ be a symmetric-key encryption scheme and let A be a probabilistic algorithm (called an adversary). In the find stage, adversary A outputs two distinct messages, x_0, x_1 , and some information, s, to use in the next mode, then, in the guess stage, takes the encryption of x_b where $b \leftarrow_R \{0,1\}$ and the above information, and tries to guess b. The advantage of A is meant by how well she can guess the value b. In our definition, unlike [5], A doesn't have any encryption oracle.

Definition 9. [Find-Guess] Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathsf{KSPC}, \mathsf{MSPC})$ be a symmetric-key encryption scheme and let A be an adversary. For $k \in \mathbb{N}$, define the advantage of A, by $\mathrm{Adv}_{A,\Pi}^{\mathrm{fg}}(k) =$

$$\begin{aligned} 2 \cdot \Pr[a \leftarrow_R \mathsf{KSPC}(k); (x_0, x_1, s) \leftarrow A(\mathrm{find}); b \leftarrow_R \{0, 1\}; \\ y &= \mathcal{E}_a(x_b) : A(\mathrm{guess}, s, y) = b] - 1. \end{aligned}$$

We require that, for (x_0, x_1) that A outputs, $x_0 \neq x_1$ and $x_0, x_1 \in MSPC$.

We say that adversary $A(t,\epsilon)$ -breaks Π in the sense of FG in the random oracle model if A runs in at most time t and achieves $\operatorname{Adv}_{A,\Pi}^{\operatorname{fg}}(k) \geq \epsilon$. We say Π is (t,ϵ) -secure in the sense of FG if there is no adversary that (t,ϵ) -breaks Π in that sense.

5 Security

This section shows the concrete security reduction.

5.1 Basic Conversion

Lemma 10. (Chosen-Plaintext Security) Suppose Π^{asym} is (t_1, ϵ_1) -secure in the sense of OWE and Π^{sym} is (t_2, ϵ_2) -secure in the sense of FG. Let l_1 and l_2 be the sizes of MSPC^{asym} and MSPC^{sym}, respectively. Then Π^{hy} is $(t, q_g, q_h, \epsilon_0)$ -secure in the sense of IND-CPA in the random oracle model, where

$$t = \min(t_1, t_2) - O(l_1 + l_2) \quad and \quad \epsilon_0 = 2(q_g + q_h)\epsilon_1 + \epsilon_2.$$

The proof is described in Appendix.

Lemma 11. (Knowledge Extractor) Suppose $\Pi^{\operatorname{asym}}$ is γ -uniform and (t_1, ϵ_1) -secure in the sense of OWE. Suppose Π^{sym} is (t_2, ϵ_2) -secure in the sense of FG. Let l_1 and l_2 be the sizes of MSPC^{asym} and MSPC^{sym}, respectively. Suppose B is a (q_g, q_h, q_e) -adversary for Π^{hy} . Then, there exist a (t, λ) -knowledge extractor, K, for B such that

$$t = O((q_g + q_h) \cdot (l_1 + l_2)) \quad and \quad \lambda = 1 - q_e \cdot \epsilon_1 - 2\epsilon_2 - \gamma - 2^{-l_2}.$$

The proof is described in Appendix.

Next is our main theorem. We omit the proof here since, due to the result of [4], it is straightforwards, provided lemmas 10 and 11 hold true (see [4]) (The proof will be described in the full paper version).

Theorem 12. (Chosen-Ciphertext Security) Suppose $\Pi^{\operatorname{asym}}$ is γ -uniform and (t_1, ϵ_1) -secure in the sense of OWE. Suppose Π^{sym} is (t_2, ϵ_2) -secure in the sense of FG. Let l_1 and l_2 be the sizes of MSPC^{asym} and MSPC^{sym}, respectively. Then Π^{hy} is $(t, q_g, q_h, q_d, \epsilon)$ -secure in the sense of IND-CCA in the random oracle model where

$$\begin{split} t &= \min(t_1, t_2) - O((q_g + q_h) \cdot (l_1 + l_2)) \quad and \\ \epsilon &= (2(q_g + q_h)\epsilon_1 + \epsilon_2 + 1)(1 - 2\epsilon_1 - 2\epsilon_2 - \gamma - 2^{-l_2})^{-q_d} - 1. \end{split}$$

5.2 A Variant: Symmetric Encryption is One-Time Padding

When a symmetric encryption, Π^{sym} , is one-time padding, we can relax the security condition.

Here define a symmetric encryption scheme by $\mathcal{E}_a^{\mathrm{sym}}(m) = a \oplus m$ (and $\mathcal{D}_a^{\mathrm{sym}}(c) = a \oplus c$). Define the key space $\mathrm{KSPC^{\mathrm{sym}}} = \{0,1\}^{l_2}$ and the message space $\mathrm{MSPC^{\mathrm{sym}}} = \{0,1\}^{l_2}$. Then $G: \mathrm{MSPC^{\mathrm{asym}}} \to \{0,1\}^{l_2}$ and $H: \mathrm{MSPC^{\mathrm{asym}}} \times \{0,1\}^{l_2} \to \{0,1\}^{l_2}$.

In $\Pi^{\text{hy}} = (\mathcal{K}^{\text{hy}}, \mathcal{E}^{\text{hy}}, \mathcal{D}^{\text{hy}}, \text{COINS}^{\text{hy}}, \{0, 1\}^{l_2})$, the encryption of a plaintext m is then

$$\mathcal{E}_{nk}^{\text{hy}}(m) = \mathcal{E}_{nk}^{\text{asym}}(\sigma; H(\sigma, m)) \mid\mid G(\sigma) \oplus m.$$

Then we can show the following results:

Corollary 13. (Knowledge Extractor) Suppose Π^{asym} is γ -uniform. Let l_1 be the size of MSPC^{asym}. Suppose B is a (q_g, q_h, q_e) -adversary for Π^{hy} . Then, there exist a (t, λ) -knowledge extractor, K, for B such that

$$t = O((q_q + q_h) \cdot (l_1 + l_2))$$
 and $\lambda = 1 - \gamma - 2^{-l_2}$.

Theorem 14. (Chosen-Ciphertext Security) Suppose Π^{asym} is γ -uniform and (t_1, ϵ_1) -secure in the sense of OWE. Let l_1 be the size of MSPC^{asym}. Then Π^{hy} is $(t, q_g, q_h, q_d, \epsilon)$ -secure in the sense of IND-CCA in the random oracle model where

$$t = t_1 - O((q_g + q_h) \cdot (l_1 + l_2)) \quad and$$

$$\epsilon = (2(q_g + q_h)\epsilon_1 + 1)(1 - \gamma - 2^{-l_2})^{-q_d} - 1.$$

These proofs will be written in the full paper version.

6 Implementation

6.1 Implementation for the ElGamal Encryption Scheme

Let G_q be an Abelian group of the order q, where the group law is expressed by addition. We assume that the Diffie-Hellman problem defined over the underlying group is difficult. Let \mathbf{g} be a generator of G_q , and (\mathbf{y},x) denote a pair of a public and secret keys such that $\mathbf{y} = x \cdot \mathbf{g}$, where $x \in \mathbb{Z}/q\mathbb{Z}$. Let $[0,1,\ldots,q-1] \subseteq \mathtt{MSPC}^{\mathrm{sym}}$ be an encoding of G_q . Let $\mathtt{hash}_1:[0,1,\ldots,q-1] \longrightarrow \mathtt{KSPC}^{\mathrm{sym}}$ and $\mathtt{hash}_2:[0,1,\ldots,q-1] \times \mathtt{MSPC}^{\mathrm{sym}} \longrightarrow [0,1,\ldots,q-1]$ be hash functions.

Encryption
$$\mathcal{E}_{pk}^{\text{hy}}(m)$$
 Decryption $\mathcal{D}_{sk}^{\text{hy}}(\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3})$
$$\sigma \leftarrow_R [0, 1, \dots, q-1], \\ \mathbf{c_1} := \sigma + \text{hash}_2(\sigma, m) \cdot \mathbf{y}, \\ \mathbf{c_2} := \text{hash}_2(\sigma, m) \cdot \mathbf{g}, \\ \mathbf{c_3} := \mathcal{E}_{\text{hash}_1(\sigma)}^{\text{sym}}(m).$$
 If $\mathbf{c_1} = \hat{\sigma} + \text{hash}_2(\hat{\sigma}, \hat{m}) \cdot \mathbf{y} \\ \text{then } m := \hat{m} \text{ else } m := \bot.$ return $(\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3})$.

Note 15. Let k = |q| be a security parameter. The ElGamal encryption primitive, associated with 1^k , is 2^{-k} -uniform.

For an application to the elliptic curve encryption system, see [15].

6.2 Implementation for the Okamoto-Uchiyama Scheme

Let $n=p^2q$ be a large positive integer such that p and q are both primes of the same size, i.e., |p|=|q|=k+1. Let $\mathbb{Z}/n\mathbb{Z}$ and $(\mathbb{Z}/n\mathbb{Z})^{\times}$ be the integer ring modulo n and the multiplicative group of $\mathbb{Z}/n\mathbb{Z}$. We assume that the factoring of n is difficult. For $g,h_0\in(\mathbb{Z}/n\mathbb{Z})^{\times}$, let $g_p=g^{p-1}$ mod p^2 and let $h=h_0^n$ mod n. Define $L(x)=\frac{x-1}{p}$ for $x\in\mathbb{Z}$. Let $\mathrm{hash}_1:\{0,1\}^k\longrightarrow\mathrm{KSPC^{\mathrm{sym}}}$ and $\mathrm{hash}_2:\{0,1\}^k\times\mathrm{MSPC^{\mathrm{sym}}}\longrightarrow\{0,1\}^{3k}$ be hash functions. Let pk=(n,g,h,k) be the public-key and let $sk=(p,q,g_p,L(g_p))$ be the secret-key.

Encryption
$$\mathcal{E}_{pk}^{\text{hy}}(m)$$
 Decryption $\mathcal{D}_{sk}^{\text{hy}}(c_1, c_2)$
$$\sigma \leftarrow_R \{0, 1\}^k, c_1 := g^{\sigma} h^{\text{hash}_2(\sigma, m)} \mod n, c_2 := \mathcal{E}_{\text{hash}_1(\sigma)}^{\text{sym}}(m).$$

$$c_1 := g^{\sigma} h^{\text{hash}_2(\sigma, m)} \mod n, c_2 := \mathcal{E}_{\text{hash}_1(\sigma)}^{\text{sym}}(m).$$

$$f(c_1) := c_1^{p-1} \mod p^2, c_2 := \mathcal{L}(c_{1,p}) \cdot \mathcal{L}(g_p)^{-1} \mod p, c_3 := \mathcal{L}(c_{1,p}) \cdot \mathcal{L}(g_p)^{-1} \mod p, c_4 := \mathcal{L}(c_1) := g^{\sigma} h^{\text{hash}_2(\sigma, m)} \pmod n, c_4 := g^{\sigma} h^{\text{hash}_2(\sigma, m)} \pmod n, c_5 := g^{\sigma} h^{\text{hash}_2(\sigma, m)}$$

Note 16. This Okamoto-Uchiyama encryption primitive, associated with 1^k , is 2^{-2k} -uniform $(2^{2k} \approx \phi(pq))$.

For more detailed information of this implementation, see [16].

7 Notes on γ -uniformity

As described in theorem 12, the security (IND-CCA) of our proposed scheme depends only on the security of the asymmetric and symmetric encryption primitives (OWE and FG, respectively) and γ -uniformity of the asymmetric encryption primitive. As γ increases to 1 (the variants of the encryption decrease), the security parameter ϵ , described in theorem 12, become larger (become worse). Since γ is evaluated at the worst point in the message space, one should choose MSPCasym very carefully, not including a singular point at which γ gets very close to 1. If $\gamma=1$, then ϵ doesn't make sense any more (e.g., the asymmetric encryption primitive is a deterministic encryption scheme). Thus this conversion can't directly apply to such an asymmetric encryption scheme as the RSA encryption primitive. However, one can easily modify it and decrease parameter γ , as follows:

$$\hat{\mathcal{E}}_{pk}^{\mathrm{asym}}(m;(r||r')) = \mathcal{E}_{pk}^{\mathrm{asym}}(m;r) \ || \ r'$$

where $\mathcal{E}_{pk}^{\text{asym}}(m;r)$ is an encryption algorithm in an asymmetric encryption scheme. Clearly the new asymmetric encryption scheme still meets the security notion of OWE if the original one meets the security notion. In particular, suppose the asymmetric encryption primitive is a OWTP and the symmetric encryption primitive is one-time padding. We then have, for a OWTP, f,

$$\mathcal{E}_{nk}^{\text{hy}}(m) = (f(\sigma)||H(\sigma,m)) \quad || \quad (G(\sigma) \oplus m).$$

This coincides with the encryption scheme presented in [2] as a chosen-ciphertext secure encryption scheme (IND-CCA).

Acknowledgment

We would like to greatly thank Phil Rogaway for his invaluable support in revising our manuscript. We also wish to thank anonymous reviewers for useful comments.

References

- M. Abdalla, M. Bellare and P. Rogaway, "DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem," Submission to IEEE P1363.
- 2. M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," Proc. of the First ACM Conference on Computer and Communications Security, pp.62–73.
- 3. M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption—How to encrypt with RSA" Advances in Cryptology –EUROCRYPT'94.
- M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes" Advances in Cryptology
 –CRYPTO'98.

- M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation", Proceedings of FOCS97, IEEE, 1997.
- M. Blum, and S. Goldwasser, "An efficient probabilistic public-key encryption scheme which hides all partial information", Proceeding of CRYPTO'84, LNCS 196, Springer-Verlag, pp.289-299 (1985).
- 7. Canetti, R., Goldreich, O. and Halevi, S.: The Random Oracle Methodology, Revisited, Proc. of STOC, ACM Press, pp.209-218 (1998).
- 8. R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen message attack", Advances in Cryptology –CRYPTO'98, Springer-Verlag, 1998.
- 9. I. Damgård, "Towards practical public key systems secure against chosen ciphertext attacks", Advances in Cryptology –CRYPTO'91, pp.445–456, Proceedings, Lecture Notes in Computer Science No. 576, Springer-Verlag, 1992.
- D. Dolev and C. Dwork and M. Naor, "Non-malleable cryptography", Proceeding of STOC91, pp 542–552.
- T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory, IT-31, 4, pp.469–472, 1985.
- 12. Fujisaki, E. and Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost, Proc. of PKC'99, LNCS, Springer-Verlag (1999).
- S. Goldwasser, and S. Micali, "Probabilistic Encryption", JCSS, vol.28, pp.270– 299, 1984.
- M. Naor, and M. Yung "Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks", Proceeding of the 22nd Annual Symposium on Theory of Computing, ACM (STOC), pp.427-437, 1990
- T. Okamoto, E. Fujisaki and H. Morita, "PSEC: Provably Secure Elliptic Curve Encryption Scheme", Submission to IEEE P1363a, March 1999.
- 16. T. Okamoto, S. Uchiyama and E. Fujisaki, "EPOC: Efficient Probabilistic Public-Key Encryption", Submission to IEEE P1363a, November 1998.
- 17. T. Okamoto, and S. Uchiyama, "A New Public-Key Cryptosystem as Secure as Factoring", Advances in Cryptology –EUROCRYPT'98, Springer-Verlag, 1998.
- 18. M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", MIT Technical Report, MIT/LCS/TR-212, 1979.
- C. Rackoff and D.R. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack", Advances in Cryptology -CRYPTO91, pp.433-444, Proceedings, Lecture Notes in Computer Science No. 576, Springer-Verlag, 1992.
- R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of ACM, 21, 2, pp.120-126, 1978.
- V. Shoup, and R. Gennaro, "Securing Threshold Cryptosystems against Chosen Ciphertext Attack", Advances in Cryptology –EUROCRYPT'98, Springer-Verlag, 1998.
- Y. Tsiounis and M. Yung, "On the Security of ElGamal based Encryption", P-KC'98, January, 1998.
- 23. Y. Zheng and J. Seberry, "Practical Approaches to Attaining Security Against Adaptively Chosen Ciphertext Attacks", Advances in Cryptology -CRYPTO'92, pp.292-304, Proceedings, Lecture Notes in Computer Science No. 740, Springer-Verlag, 1992.

A Proof of Lemma 10

Suppose for contradiction that there exists an adversary, A_0 , that $(t, q_g, q_h, \epsilon_0)$ -breaks Π^{hy} in MSPC^{asym} in the sense of IND-CPA in the random oracle model. We can then show that there exist adversaries, B_1 and B_2 , such that B_1 (t_1, ϵ_1) -breaks Π^{asym} in MSPC^{asym} in the sense of OWF and B_2 (t_2, ϵ_2) -breaks Π^{sym} in the sense of FG, where $t_1 = t + O(l_1 + l_2)$, $t_2 = t + O(l_1 + l_2)$, and $\epsilon_0 \leq 2 \cdot (q_g + q_h) \cdot \epsilon_1 + \epsilon_2$.

We show how to construct B_1 and B_2 , which take advantage of A_0 as an oracle. Here, when utilizing A_0 as an oracle, B_1 and B_2 should make, by themselves, the answers for A_0 's queries instead of random oracles, G and H, i.e., B_1 and B_2 have to simulate the random oracles. We describe the procedure, which is shared with B_1 and B_2 , of the simulation in the following.

[The procedure of making \mathcal{T}_G and \mathcal{T}_H] Recall that G and H are ideal random functions specified by $G: MSPC^{asym} \to KSPC^{sym}$, and $H: MSPC^{asym} \times MSPC^{sym} \to COINS^{asym}$. At first we prepare empty lists, \mathcal{T}_G and \mathcal{T}_H and a counter, count $\leftarrow 0$.

- How to make \mathcal{T}_G For a query, σ , if it hasn't been entered as an entry in \mathcal{T}_G , choose g random and uniformly from KSPC^{sym}, answer g to adversary A_0 , increase count by 1, set $(\sigma_{\text{count}}, g_{\text{count}}) := (\sigma, g)$, and put it on \mathcal{T}_G , otherwise answer g_i such that $\sigma_i = \sigma$ and $(\sigma_i, g_i) \in \mathcal{T}_G$.
- How to make \mathcal{T}_H For a query, (σ, m) , if it hasn't been entered as an entry in \mathcal{T}_H , choose h random and uniformly from COINS^{asym}, answer h to adversary A_0 , increase count by 1, set $(\sigma_{\mathtt{count}}, m_{\mathtt{count}}, h_{\mathtt{count}}) := (\sigma, m, h)$, and put it on \mathcal{T}_H , otherwise answer h_i such that $(\sigma_i, m_i) = (\sigma, m)$ and $(\sigma_i, m_i, g_i) \in \mathcal{T}_H$.

[Adversary B_1] We explain the specification of adversary B_1 . Recall that B_1 is an algorithm that on input (pk,y) outputs some string, and the advantage is specified by $Adv_{B_1,H^{\mathrm{asym}}}^{\mathrm{cpa}}(k) = \Pr[(pk,sk) \leftarrow \mathcal{K}^{\mathrm{asym}}(1^k); x \leftarrow \mathtt{MSPC}^{\mathrm{asym}}; y \leftarrow \mathcal{E}_{pk}^{\mathrm{asym}}(x) : B_1(pk,y) = \mathcal{D}_{sk}^{\mathrm{asym}}(y)].$ B_1 runs A_0 as follows:

- **Step 1** Input pk to A_0 (originally inputted to B_1) and run A_0 in the find mode. If A_0 asks oracles, G and H, then follow the above procedure. Finally A_0 would output (x_0, x_1, s) .
- **Step 2** Choose $b \leftarrow_R \{0,1\}$ and $\hat{g} \leftarrow_R \text{KSPC}^{\text{sym}}$. Then set $c_1 := y$ and $c_2 := \mathcal{E}_{\hat{g}}^{\text{sym}}(x_b)$.
- **Step 3** Input $(x_0, x_1, s, (c_1, c_2))$ to A_0 and run A_0 in the guess mode. If A_0 asks oracles, G and H, then follow the above procedure. After asking at most $(q_g + q_h)$ queries to the random oracles or running in at most time t, A_0 would output bit b'. However if it is still running, abort it.
- **Step 4** Choose a number $i \leftarrow_R \{1, \dots, \text{count}\}$ and output σ_i as the answer to y (originally inputted to B_1).

[Adversary B_2] We now describe the specification of adversary B_2 . Recall that B_2 is an algorithm that has two modes: At first in the find mode it executes and outputs two distinct messages and some information used

in the next mode. In the guess mode, it runs on input ciphertext as well as the above information, and outputs one bit. The advantage is specified by $Adv_{B_2,\Pi^{\mathrm{sym}}}^{\mathrm{fg}}(k) = 2 \cdot \Pr[a \leftarrow \mathsf{KSPC^{\mathrm{sym}}}; (x_0, x_1, s) \leftarrow B_2(\mathrm{find}); b \leftarrow_R \{0, 1\}; y = \mathcal{E}_a^{\mathrm{sym}}(x_b) : B_2(\mathrm{guess}, x_0, x_1, s, y) = b] - 1$. B_2 runs A_0 as follows:

- **Step 1** (B_2 is set in the find mode.) Run $\mathcal{K}^{\operatorname{asym}}$ on input 1^k and let K output (pk, sk). Do the same thing as in the first step in the case of adversary B_1 . Finally A_0 outputs (x_0, x_1, s) , then output (x_0, x_1, s) as his own output.
- **Step 2** (B_2 is inputted $y = \mathcal{E}_a^{\text{sym}}(x_b)$ where $a \leftarrow_R \text{KSPC}^{\text{sym}}$, $b \leftarrow_R \{0,1\}$, and enter the guess mode.) Choose $\sigma \leftarrow_R \text{MSPC}^{\text{asym}}$ and $\hat{h} \leftarrow_R \text{COINS}^{\text{asym}}$. Then set $c_1 := \mathcal{E}^{\text{asym}}(\sigma, \hat{h})$ and $c_2 := y$.
- **Step 3** (B_2 is still in the guess mode.) Do the same thing as in the third step in the case of adversary B_1 .
- **Step 4** (B_2 is still in the guess mode.) Finally, A_0 outputs b'. Then output b' as his own answer.

From the specifications of B_1 and B_2 , their running times are $t + O(l_1 + l_2)$. Here we define, for shorthand, the following experiment:

$$\begin{split} \operatorname{Ask} A_0 &= [\ A_0 \ \operatorname{asks} \ G \ \operatorname{or} \ H \ \operatorname{a} \ \operatorname{query} \ \operatorname{that} \ \operatorname{includes} \ \mathcal{D}_{sk}^{\operatorname{asym}}(c_1).] \\ \operatorname{Succ} A_0 &= [G, H \leftarrow \varOmega; (pk, sk) \leftarrow \mathcal{K}^{\operatorname{asym}}(1^k); (x_0, x_1, s) \leftarrow A_0{}^{G, H}(\operatorname{find}, pk); \\ b \leftarrow_R \{0, 1\}; y \leftarrow \mathcal{E}_{pk}^{\operatorname{hy}}(x_b) : A_0{}^{G, H}(\operatorname{guess}, x_0, x_1, s, y) = b] \\ \operatorname{Succ} B_1 &= [(pk, sk) \leftarrow \mathcal{K}^{\operatorname{asym}}(1^k); x \leftarrow \operatorname{MSPC}^{\operatorname{asym}}; y \leftarrow \mathcal{E}_{pk}^{\operatorname{asym}}(x) : \\ B_1(pk, y) &= \mathcal{D}_{sk}^{\operatorname{asym}}(y)] \\ \operatorname{Succ} B_2 &= [sk_2 \leftarrow_R \operatorname{KSPC}^{\operatorname{sym}}; (x_0, x_1, s) \leftarrow B_2(\operatorname{find}); b \leftarrow_R \{0, 1\}; \end{split}$$

In addition, let $p_0 = \Pr[AskA_0]$, so we can write

$$\Pr[\operatorname{Succ} A_0] = \Pr[\operatorname{Succ} A_0 | \operatorname{Ask} A_0] \cdot p_0 + \Pr[\operatorname{Succ} A_0 | \overline{\operatorname{Ask} A_0}] \cdot (1 - p_0),$$

$$\Pr[\operatorname{Succ} B_2] = \Pr[\operatorname{Succ} B_2 | \operatorname{Ask} A_0] \cdot p_0 + \Pr[\operatorname{Succ} B_2 | \overline{\operatorname{Ask} A_0}] \cdot (1 - p_0).$$

Then, from the specification of adversaries, B_1 and B_2 , it holds

 $y = \mathcal{E}_{a}^{\text{sym}}(x_b) : A_0^{G,H}(\text{guess}, x_0, x_1, s, y) = b]$

$$\Pr[\operatorname{Succ} B_1] \ge (q_g + q_h)^{-1} \cdot p_0 \text{ and } \Pr[\operatorname{Succ} B_2] \ge \Pr[\operatorname{Succ} A_2 | \overline{\operatorname{Ask} A_0}] \cdot (1 - p_0).$$

This is because: if A_0 asks at least one query including $\mathcal{D}_{sk}^{\mathrm{asym}}(y)$ to either G or H, then B_1 can output the correct answer with probability at least $1/(q_g+q_h)$. Otherwise, $\Pr[\mathrm{Succ}A_0|\overline{\mathrm{Ask}A_0}]\cdot (1-p_0)=\Pr[\mathrm{Succ}B_2|\overline{\mathrm{Ask}A_0}]\cdot (1-p_0)$. Therefore,

$$\Pr[\operatorname{Succ} A_0] \le (q_q + q_h) \cdot \Pr[\operatorname{Succ} B_1] + \Pr[\operatorname{Succ} B_2]. \tag{1}$$

Then, from the assumption (for contradiction), we can write

$$\epsilon_0 \leq 2\Pr[\operatorname{Succ} A_0] - 1, \quad \epsilon_1 = \Pr[\operatorname{Succ} B_1], \quad \epsilon_2 = 2\Pr[\operatorname{Succ} B_2] - 1. \tag{2}$$

Hence,
$$\epsilon_0 \leq 2(q_g + q_h) \cdot \epsilon_1 + \epsilon_2$$
.

B Proof of Lemma 11

Let B be a (q_g, q_h, q_e) -adversary that, on input pk, asks queries to G and H, asks queries to the encryption oracle, $\mathcal{E}_{pk}^{\text{hy}}(\cdot)$, and finally outputs (c_1, c_2) , where $(c_1, c_2) \not\in \mathcal{Y}$. Recall we write the experiment as $(\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, y) \leftarrow B^{G,H,\mathcal{E}_{pk}^{\text{hy}}}(pk)$. The knowledge extractor, K, is an algorithm which, on input $(\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, (c_1, c_2), pk)$, outputs a string. Recall for $k \in \mathbb{N}$, Succ $_{K,B,H^{\text{hy}}}^{\text{ke}}(k) =$

$$\Pr[G, H \leftarrow \Omega; (pk, sk) \leftarrow \mathcal{K}(1^k); (\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, y) \leftarrow B^{H, \mathcal{E}_{pk}^{\text{hy}}}(pk) : \\ K(\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, y, pk) = \mathcal{D}_{sk}^{\text{hy}}(y)].$$

[Knowledge Extractor] Here, let $\mathcal{T}_G = \{(\sigma_i, g_i) | i = 1, \dots, q_g\}$ and $\mathcal{T}_H = \{(\sigma'_j, m_j, h_j) | j = 1, \dots, q_h\}$. We then give the specification of the knowledge extractor K as follows:

- **Step 1** Set two empty lists, S_1 and S_2 .
- **Step 2** Find all elements in \mathcal{T}_H such that $c_1 = \mathcal{E}_{pk}^{\operatorname{asym}}(\sigma'_j, h_j)$ and put them into list S_1 . If $S_1 = \emptyset$, then output \bot , otherwise
- **Step 3** For every (σ'_j, m_j, h_j) in S_1 , find all elements in \mathcal{T}_G such that $\sigma_i = \sigma'_j$ and put them (i.e., $(\sigma'_j, m_j, h_j)||(\sigma_i, g_i)$'s) into S_2 . If $S_2 = \emptyset$, then output \bot , otherwise
- **Step 4** Check in S_2 if there exists a $(\sigma'_j, m_j, h_j)||(\sigma_i, g_i)$ such that $c_2 = \mathcal{E}^{\text{sym}}_{q_i}(m_j)$. If it exists in S_2 , then output m_j otherwise output \perp .

This protocol runs in time $O((q_q + q_h)k)$.

Next we examine the advantage of the knowledge extractor. We define the following events:

- Inv is true if there exists $(c_1^*, c_2^*) \in \mathcal{Y}$ and $(\sigma_i, g_i) \in \mathcal{T}_G$ or $(\sigma_j, m_j, h_j) \in \mathcal{T}_H$ such that $\sigma_i = D_{sk}^{\operatorname{asym}}(c_1^*)$ or $\sigma_j = D_{sk}^{\operatorname{asym}}(c_1^*)$.
- $-p(S_1)$ is true if $S_1 \neq \emptyset$.
- $-p(S_2)$ is true if $S_2 \neq \emptyset$.
- Find is true if there exists a $(\sigma'_j, m_j, h_j)||(\sigma_i, g_i)$ in S_2 such that $c_2 = \mathcal{E}^{\text{sym}}_{g_i}(m_j)$.
- Fail is true if "the output of knowledge extractor K" $\neq \mathcal{D}_{sk}^{\text{hy}}(c_1, c_2)$.

We further define the following events:

'1' = Inv.
'00' =
$$\neg$$
Inv $\land \neg p(S_1)$.
'010' = \neg Inv $\land p(S_1) \land \neg p(S_2)$.
'0110' = \neg Inv $\land p(S_1) \land p(S_2) \land \neg$ Find.
'0111' = \neg Inv $\land p(S_1) \land p(S_2) \land$ Find.

Then the following equation holds

$$\begin{split} \Pr[\mathrm{Fail}] &= \Pr[\mathrm{Fail}|1] \cdot \Pr[1] + \Pr[\mathrm{Fail}|00] \cdot \Pr[00] + \Pr[\mathrm{Fail}|010] \cdot \Pr[010] + \\ &\quad \Pr[\mathrm{Fail}|0110] \cdot \Pr[0110] + \Pr[\mathrm{Fail}|0111] \cdot \Pr[0111]. \end{split}$$

Hence,

$$\Pr[\mathrm{Fail}] \leq \Pr[1] + \Pr[\mathrm{Fail}|00] + \Pr[\mathrm{Fail}|010] + \Pr[\mathrm{Fail}|0110] + \Pr[\mathrm{Fail}|0111].$$

Here we can easily find that Pr[Fail|0110] = Pr[Fail|0111] = 0. In addition, we claim the following inequalities hold true:

Claim.
$$\Pr[1] \leq q_e \cdot \epsilon_1$$
.

Proof. Remember that the interaction between adversary B and encryption oracle $\mathcal{E}_{pk}^{\text{hy}}(\cdot)$: When B make access to $\mathcal{E}_{pk}^{G,H}$ with query m, $\mathcal{E}_{pk}^{\text{hy}}$ takes random coins $\sigma \leftarrow_R \text{COINS}^{\text{asym}}$ and answer to B with $(\mathcal{E}_{pk}^{\text{asym}}(\sigma, H(\sigma, m)) \mid\mid \mathcal{E}_{G(\sigma)}^{\text{sym}}(m))$. Bmakes at most q_e queries to $\mathcal{E}_{pk}^{\text{hy}}$. Therefore, $\Pr[1] = \Pr[\text{Inv}] \leq q_e \cdot \epsilon_1$.

Therefore,
$$\Pr[1] = \Pr[\text{Inv}] \leq q_e \cdot \epsilon_1$$
.

Claim. $Pr[Fail|00] < \gamma$

Proof. Given the event 00, we can identify B with an adversary B' which on input pk outputs a string, y, to guess the random coins in y. The advantage of B' is $Adv_{B'} \Pi^{asym}(k) =$

$$\begin{split} \Pr[(pk,sk) \leftarrow \mathcal{K}^{\text{asym}}(1^k); h \leftarrow_R \text{COINS}^{\text{asym}}; y \leftarrow B'(pk); \\ x \leftarrow \mathcal{D}_{sk}^{\text{asym}}(y) : y = \mathcal{E}_{pk}^{\text{asym}}(x;h)] \end{split}$$

Then

$$\begin{split} \Pr[\mathrm{Fail}|\mathsf{00}] & \leq \mathrm{Adv}_{B',H^{\mathrm{asym}}}(k) \\ & = \Pr[(pk,sk) \leftarrow \mathcal{K}^{\mathrm{asym}}(1^k); y \leftarrow B'(pk); x \leftarrow \mathcal{D}_{sk}^{\mathrm{asym}}(y); \\ & h \leftarrow_R \mathtt{COINS}^{\mathrm{asym}}: y = \mathcal{E}_{pk}^{\mathrm{asym}}(x;h)]. \end{split}$$

Recall, for $(pk, sk) \in \mathcal{K}^{asym}(1^k)$, $x \in MSPC^{asym}$ and $y \in \{0, 1\}^*$,

$$\gamma(x,y) = \Pr[h \leftarrow_R \mathtt{COINS^{asym}} : y = \mathcal{E}^{\mathrm{asym}}_{pk}(x;h)] \leq \gamma.$$

Hence
$$\Pr[\text{Fail}|00] < \gamma$$
.

Claim. $Pr[Fail|010] < 2\epsilon_2 + 2^{-l_2}$.

Proof. Given the event 010, we can identify B with an adversary B' that outputs a pair of strings to guess the secret-key of Π^{sym} . The advantage of B' is $\operatorname{Adv}_{B',\Pi^{\operatorname{sym}}}(k) =$

$$\Pr[g \leftarrow_R \mathtt{KSPC^{sym}}; (x, y) \leftarrow B'(\mathrm{find}) : y = \mathcal{E}_g^{\mathrm{sym}}(x)].$$

Since the event, Fail 010, means that B' outputs valid (x, y) (extractor Koutputs \perp), $\Pr[\text{Fail}|010] = \text{Adv}_{B',\Pi^{\text{sym}}}(k)$.

[Adversary A] Suppose there exists B' with $\delta := \operatorname{Adv}_{B',\Pi^{\text{sym}}}(k)$. We then construct adversary A against Π^{sym} as follows:

- **Step 1** (A starts in the find mode.) Run B' and finally B' outputs (x', y').
- **Step 2** Choose $b' \leftarrow_R \{0,1\}$ and $x'' \leftarrow_R \{0,1\}^{|x'|}$. Then set $x_{b'} := x'$ and $x_{\bar{b}'} := x''$ (where \bar{b}' denotes the complement of b'). Finally output (x_0, x_1) .
- **Step 3** (A is inputted $y = \mathcal{E}_g^{\mathrm{sym}}(x_b)$ where $g \leftarrow_R \mathrm{KSPC^{\mathrm{sym}}}$ and $b \leftarrow_R \{0,1\}$ and enters the guess mode.) If y = y' then output b' else flip a coin again and output the result (namely, after $b'' \leftarrow_R \{0,1\}$ output b'').

Define Succ A =

$$[g \leftarrow_R \mathsf{KSPC^{sym}}; (x_0, x_1) \leftarrow A(\mathsf{find}); b \leftarrow \{0, 1\}; y = \mathcal{E}_g^{\mathsf{sym}}(x_b) : A(\mathsf{guess}, s, y) = b].$$

Then,

$$\Pr[\operatorname{Succ} A] = \Pr[\operatorname{Succ} A|y = y'] \cdot \Pr[y = y'] + \Pr[\operatorname{Succ} A|y \neq y'] \cdot \Pr[y \neq y'].$$

Define $X=(x,y),\ Q_X=\Pr_B[(x,y)\leftarrow B'(\text{find}],\ \text{and}\ P_X=\frac{\#\{g|y=\mathcal{E}_g^{\text{sym}}(x)\}}{\#\text{KSPC}^{\text{Sym}}}.$ We then have

$$\delta = \sum_{X} Q_X P_X.$$

To evaluate $\Pr[\operatorname{Succ} A]$, let X' := (x', y') and X'' := (x'', y'). Here note that $Q_{X'} = Q_{X''}$. First we evaluate the conditional probability of $\Pr[\operatorname{Succ} A]$ given X', X'' (outputs of B and A); we write the probability as $\Pr_{X', X''}[\operatorname{Succ} A]$.

$$\Pr_{X',X''}[y = y'] = \frac{1}{2} \cdot \Pr_{X',X''}[y' = \mathcal{E}_g^{\text{sym}}(x')] + \frac{1}{2} \cdot \Pr_{X',X''}[y' = \mathcal{E}_g^{\text{sym}}(x'')]$$

$$= \frac{1}{2} (P_{X'} + P_{X''}),$$

$$\Pr_{X',X''}[\operatorname{Succ} A|y=y'] = \frac{\#\{g \mid y' = \mathcal{E}_g^{\operatorname{sym}}(x')\}}{\#\{g \mid y' = \mathcal{E}_g^{\operatorname{sym}}(x') \vee y' = \mathcal{E}_g^{\operatorname{sym}}(x'')\}} = \frac{P_{X'}}{P_{X'} + P_{X''}},$$

 $\Pr_{X',X''}[\operatorname{Succ} A|y \neq y'] = \frac{1}{2}$, and $\Pr_{X',X''}[y \neq y'] = 1 - \Pr_{X',X''}[y = y'] = 1 - \frac{1}{2}(P_{X'} + P_{X''})$. Therefore,

$$\Pr_{X',X''}[\operatorname{Succ} A] = \frac{1}{2} + \frac{1}{4}(P_{X'} - P_{X''}).$$

We then evaluate $\Pr[\operatorname{Succ} A]$ (Here note that $Q_{X'}$ is taken over B''s coin flips and $\Pr[x'' \leftarrow A(\operatorname{find})] = 2^{-l_2}$ is taken over A's coin flips.)

$$\begin{aligned} \Pr[\operatorname{Succ} A] &= \left(\frac{1}{2}\right)^{l_2} \sum_{X'} \sum_{X''} Q_{X'} \left(\frac{1}{2} + \frac{1}{4} (P_{X'} - P_{X''})\right) \\ &= \frac{1}{2} + \frac{1}{4} \delta - \frac{1}{4} \sum_{X'} \sum_{X''} Q_{X'} P_{X''}, \end{aligned}$$

since $\sum_{X'} Q_{X'} P_{X'} = \delta$, $\sum_{X'} Q_{X'} = 1$, and $\sum_{x''} \left(\frac{1}{2}\right)^{l_2} = 1$.

By combining with the assumption that Π^{sym} is (t_2, ϵ_2) -secure, i.e., $\Pr[\text{Succ} A] \leq \frac{1}{2} + \frac{\epsilon_2}{2}$ (that is, $\frac{1}{2} + \frac{1}{4}\delta - \frac{1}{4}\sum_{X'}\sum_{X''}Q_{X'}P_{X''} \leq \frac{1}{2} + \frac{\epsilon_2}{2}$), we have the following inequality,

$$\delta \le 2\epsilon_2 + \sum_{X'} \sum_{X''} Q_{X'} P_{X''},$$

We then evaluate $\sum_{X'} \sum_{X''} Q_{X'} P_{X''}$.

$$\sum_{X'} \sum_{X''} Q_{X'} P_{X''} = \sum_{X'} Q_{X'} \Big(\Big(\frac{1}{2} \Big)^{l_2} \sum_{x''} \frac{\# \{g \mid y' = \mathcal{E}_g^{\mathrm{sym}}(x'')\}}{\# \mathrm{KSPC}^{\mathrm{sym}}} \Big).$$

Since $\{g \mid y' = \mathcal{E}_g^{\mathrm{sym}}(x_1'')\}$ should be disjoint from $\{g \mid y' = \mathcal{E}_g^{\mathrm{sym}}(x_2'')\}$ (with $x_1'' \neq x_2''$) in order to uniquely decrypt y' with key $g, \sum_{x''} \#\{g \mid y' = \mathcal{E}_g^{\mathrm{sym}}(x'')\} \leq \#\mathtt{KSPC}^{\mathrm{sym}}$. Hence,

$$\sum_{X'} \sum_{X''} Q_{X'} P_{X''} \le \left(\frac{1}{2}\right)^{l_2} \sum_{X'} Q_{X'} = \left(\frac{1}{2}\right)^{l_2}.$$

Thus, $(\delta =) \operatorname{Adv}_{B',\Pi^{\text{sym}}}(k) \leq 2\epsilon_2 + 2^{-l_2}$.

From the claims above, $\Pr[\text{Fail}] \leq q_e \cdot \epsilon_1 + \gamma + 2\epsilon_2 + 2^{-l_2}$. Therefore,

$$\lambda = 1 - \Pr[\text{Fail}] \ge 1 - (q_e \cdot \epsilon_1 + 2\epsilon_2 + 2^{-l_2} + \gamma).$$

Auditable, Anonymous Electronic Cash

(Extended Abstract)

Tomas Sander and Amnon Ta-Shma

International Computer Science Institute 1947 Center Street, Berkeley, CA 94704, USA {sander,amnon}@icsi.berkeley.edu

Abstract. Most anonymous, electronic cash systems are signature-based. A side effect of this is that in these systems the bank has the technical ability to issue unreported, valid money. It has been noticed in the past that this may lead to a disaster if the secret key of the bank is compromised. Furthermore, the above feature prevents any effective monitoring of the system.

In this paper we build a fully anonymous, auditable system, by constructing an electronic cash system that is signature-free, and where the bank needs to have no secret at all. The security of the system relies instead on the ability of the bank to maintain the integrity of a public database. Our system takes a completely new direction for meeting the above requirements, and, in particular, it is the first to do so without the necessity of making individual transactions potentially traceable: payers enjoy unconditional anonymity for their payment transactions. The system is theoretically efficient but not yet practical.

Keywords: electronic cash, anonymity.

1 Introduction

Payment systems should be safe and sound. When we deposit money into a bank we believe the bank will honor its commitments to us, and even though individual banks may run into problems (e.g., because of bad investments they make) most bank accounts are insured by national governments protecting consumers from losses. In return the banking industry is highly regulated and monitored by governmental agencies. There are at least two sides to this monitoring. One is to ensure that transactions are executed correctly. The other is to check the financial stability of the bank where it is verified that banks reasonably control their risks in issuing credits and loans, investing their assets and their operational risks (including insider crime) [32,31]. This monitoring becomes even more critical with electronic cash systems in which issuers can issue money themselves.

We call a system *auditable* if it allows to effectively control the money supply, i.e. if there can not be valid money that is not known to the auditor (we later give a more formal definition). An electronic cash system that has this property and also has a complete report of transactions (which is "standard" in monetary systems) allows monitoring the same way the banking system is monitored

today. On the other hand, if this property is violated then monitoring may not reflect the actual reality. We believe that even just for that reason auditability is important and desirable. Unfortunately, not too much attention was given to this property and there are many systems (e.g. [10]) which violate it. I.e., in such systems the issuer can issue valid money that is not reported anywhere, and thus such a system can be "technically" functioning correctly, and apparently stable, while it is in fact insolvent.

Further abuses of anonymous systems exist. Some of these exploit another bad property that may exist in an electronic cash system, and which we call rigidity. We say a system is non-rigid if any withdrawn money (in a standard or a non-standard withdrawal transaction) can be later invalidated (see a formal definition in Section 4). We argue that auditable, non-rigid systems defend against most of the major known attacks and abuses of anonymous electronic cash systems. The challenge is to build a fully anonymous yet auditable and non-rigid system. Previous solutions to the problem use electronic cash systems with "escrow" agents that can revoke the anonymity of users or coins. Thus, they compromise the anonymity feature. We take a completely different direction and present a solution that gives users full statistical anonymity.

1.1 Our Solution

In our system there is no trustee and a users' privacy can never be compromised. Furthermore, our system is not signature based, and in fact it is "secret-free"; there is no secret key the bank needs to hold (and therefore nothing that can be stolen, e.g. by insiders!). Furthermore, in our system all relevant bank actions are public and publicly verifiable, and the system is also fully auditable in the same way banks are auditable today.

The security of the system relies on the ability of the bank to maintain the integrity of a public database, and the auditability property follows from the fact that all transactions (including issuing new coins) are made public and can not be forged.

Our system also protects from blackmailing attacks against the bank; the only way a blackmailer can force the bank to issue electronic coins that will be accepted as valid payments is by forcing the bank to add some public data. The bank (and the whole world) knows which data have been added during the blackmailing, and as our system is also non-rigid the situation can be later reversed. Furthermore, even if the bank is willing to follow all terms dictated by blackmailers, there is no way for the bank to issue non-rigid coins. This leaves, for example, kidnapers with the knowledge that once they free the hostage they are left with worthless coins. Certainly also the bank knows that paying the ransom can not lead to a release of the hostage. This loose/loose scenario will strongly discourage both players to use this non-rigid system in blackmailing scenarios. Our system also has the advantage that it can remain off line after a blackmailing attack on the bank has occurred once the system has been updated.

On the conceptual level it is the first system that simultaneously guarantees unconditional user anonymity together with strong protection against the black-

mailing and the bank robbery attack, in which the bank's secret key for signing coins is compromised. Recall that previous work addressed potential criminal abuses of electronic cash systems mainly in the escrowed cash paradigm, where each user transaction is made potentially traceable by Trustee(s). As Camenisch, Piveteau and Stadler [17] put it, these type of systems "offer a compromise between the legitimate need for privacy protection and an effective prevention of misuse by criminals". Our system defends against these attacks without doing the "compromise" cited above, and this is achieved using a simple, straightforward approach.

This raises the question whether other significant system abuses like money laundering can be effectively prevented by still preserving unconditional user anonymity. That the latter is possible was shown by the authors in [36] where an amount–limited, strongly non–transferable payment system was suggested. Amount–limitedness and non-transferability assure that a money launderer can not obtain the large amounts of anonymous electronic cash that are typically involved and needed in money laundering activities, neither by withdrawing them from the bank, nor by buying them on a "black market" for anonymous electronic cash. Using the techniques developed in [36] the current system can also be made amount–limited and non-transferable. The combined system then strongly defends against blackmailing, bank robbery and money–laundering abuses while offering unconditional privacy for users in their transactions. We therefore believe that the need for escrowed cash systems should be reexamined.

1.2 Organization of the Paper

In Section 2 we describe several attacks on electronic payment systems and previous work in which they were defeated. In Section 3 we outline the ideas underlying our electronic cash system and show how it differs conceptually from previous systems offering anonymity that were blind signature based. In Section 4 we give a formal description of model and requirements for our system. In Section 5 we describe the building blocks which we use in the protocol. In Section 6 we present the protocols of our auditable anonymous system and point in Section 7 to possible directions how the efficiency of the system can be improved.

2 Previous Work

We start by describing some major attacks on anonymous electronic cash systems:

Bank robbery attack: Jakobsson and Yung [26] describe an attack where the secret key of the issuer is compromised and the thief starts counterfeiting money. The attack is especially devastating if no one will be able to detect that there is false money in the system until the amount of deposited money surpluses the amount of withdrawn money. Obviously, by that time the whole market is flooded with counterfeited money, and the system may collapse. The Group of Ten report from 1996 [14] expresses "serious concern": "one of the most significant threats

to an electronic money system would be the theft or compromising of the issuer's cryptographic keys by either an inside or an outside attacker." In the 1998 report [33] it is stated that "of direct concern to supervisory authorities is the risk of criminals counterfeiting electronic money, which is heightened if banks fail to incorporate adequate measures to detect and deter counterfeiting. A bank faces operational risk from counterfeiting, as it may be liable for the amount of the falsified electronic money balance." It further states "Over the longer term, if electronic money does grow to displace currency to a substantial degree, loss of confidence in a scheme could conceivably have broader consequences for the financial system and the economy."

Money laundering attacks: There are many possible ways to abuse an anonymous electronic cash system for money laundering (cf. e.g. [2]).

Blackmailing attack: Van Solms and Naccache [38] described a "perfect" blackmailing scenario that exploits anonymity features of (blind) signature-based electronic cash systems.

An auditor should not have to trust the issuer because an issuer can make profits from unreported money in several ways, e.g. for offering not properly backed credits or by assisting in money laundering activities (for documented cases in which financial institutions indeed assisted in unlawful activities see e.g. [1] and also [20]).

These types of attacks motivated a substantial amount of research started in [13] and [16], where electronic cash with revocable anonymity ("escrowed cash") was suggested. In, e.g. [26, 28, 15, 19, 34, 18], several systems following this approach have been described ¹. These cash systems allow a Trustee(s) to revoke the anonymity of each individual transaction. A typical revocability feature is "coin tracing" where the coin withdrawn by a user during a particular withdrawal session can be identified. This feature allows to defeat the blackmailing attack in the case a *private* user is blackmailed (as he may later ask to trace and blacklist the blackmailed coins).

Few systems of the ones mentioned above protect also against (the stronger) blackmailing attacks on the bank: a blackmailer may force the bank to enter a non–standard withdrawal protocol to withdraw coins (and thereby disable coin tracing mechanisms) or extort the bank's secret key. In the related bank robbery attack the secret key of the bank is stolen. Only the systems in [26, 19, 34, 25] prevent against these very strong latter attacks. Some of these systems require a third party involvement at withdrawal time and some can not remain off-line after such an attack had occurred.

¹ It has been pointed out before in [28] that systems like the one described in [37] are not vulnerable to several of these attacks as their security (for the bank) does not critically rely on (blind) signature techniques. Although the system [37] offers privacy to a certain extent it is not fully anonymous. We are not aware of a previous description of an electronic cash system that achieves full anonymity but does not rely on blind signatures techniques.

3 The Basic Idea

3.1 Stamps and Signatures

As stated above, whereas most previous systems offering anonymity are signature-

based and the issuer gives (blinded) signatures to coins, ours is not. Our approach and its difference to previous ones can be best understood by considering the "membership in a list" problem. In this problem a bank holds a list of values $L = \{x_1, \ldots, x_k\}$, The elements in the list "correspond" to valid coins (and will be hash values of their serial numbers). If $x \in L$ there is a short proof of membership, whereas if $x \notin L$ it is infeasible to find such a proof, i.e. only when a user has a valid coin he can give such a proof, else he can not.

Let us now focus on the membership problem. One possible solution is to keep a public list of all values in L. However, such a solution requires the verifier to be able to access a large file. Another solution is for the bank to sign each value in L and to accept an element as belonging to L iff it carries a signature, which from an abstract point of view is how systems like, e.g., [10] work. Now, however, the security of the system relies on the secrecy of the secret key of the bank and the system becomes potentially vulnerable to blackmailing attacks. A third solution for the membership problem was suggested by Merkle [27]. In Merkle's solution, all values x_1, \ldots, x_k are put into the leaves of a tree, and a hash tree is formed over the leaves using a collision resistant function h (for more details see Section 6). The root of the hash tree is made public, and the assumption is that this data is authenticated and its integrity can be maintained. A proof that x is in the list amounts to presenting a hash chain from x to the root. The collision resistant property of h then guarantees that it is infeasible to find a membership proof for an element $x \notin L$.

When the bank adds a value to the tree it "authenticates" or "stamps" the value as being valid for payment. Stamping is very similar to signing: Anyone who has access to the authenticated data (the root of the tree) can validate a stamp (by checking the hash chain leading to the root) which is similar to the public verification predicate in signature schemes. Also, it can be achieved that it is infeasible to forge neither signatures nor stamps. The key difference, from our perspective, between signatures and stamps, is that the secret/public key pair required in a signature scheme is "replaced" with authenticated public data (the root of the tree) which is exactly what we need for solving the bank robbery attack, the blackmailing attack and achieving auditability. Although the security requirements are different our system has technically some similarities to time stamping protocols [24, 8, 3] that also made use of tree based methods.

Our protocol uses Merkle's solution in a way that also allows to incorporate full anonymity, detection of double spenders and full auditability. Other features (as non-transferability and amount limitedness) can be easily added.

3.2 On the Update of Roots and Hash Chains

Let us say for the ease of explanation that a time frame lasts a minute, an hour has two minutes, a day has two hours, a week has two days, etc. At the first minute the bank creates a minute tree with all the coins issued at that minute on its leaves. At the second minute another minute tree is formed. When the second minute ends the two minute trees are combined into an hour tree by hashing the two roots of the minute trees together. During the next hour a new hour tree is used. When the second hour ends we combine the two hour trees to a day tree, and a new day tree is formed, and so on. Altogether we have a forest. Let us say that a root is alive if it is a root in the forest, i.e. it is the root of one of the last hour,day,week etc tree. If our system is to cover one hundred real years, then the system has to have about 5,000,000 roots $(100 \cdot 365 \cdot 24 \cdot 60 << 2^{26})$ and therefore 26 levels are necessary (with the bottom level covering minutes, the level above it hours etc.). In particular, at any given minute there are at most 26 live roots.

Each merchant should hold a subset of the live roots. A merchant can choose how often to update his list. If a merchant chooses to keep 20 live roots he needs to update his list every 2^6 minutes, and he can not accept coins that were issued in the last 2^6 minutes. He can choose to keep all 26 live roots (and therefore accept all issued coins) but then he needs to update his list every minute.

When a user withdraws a coin the bank sends him a hash chain from his coin to the root of the current tree, and each time the tree is combined with another tree the bank updates the chain so that it leads to the root of the combined tree. Altogether the bank sends the user 26 update messages. A payment transaction begins with the merchant sending the user the set of all live roots he knows. The user proves in a zero knowledge way that he knows a hash chain to one of the roots in the set.

We point out the following:

- The updates are independent of the actual transaction that takes place and the specific user. An update at the end of an hour should only contain the values of the roots of the last two minutes, an update at the end of a day should only contain the roots of the last two hours etc. As a result the updates can be broadcasted to every system participant.
- Each merchant can choose how often he makes the updates. The only disadvantage in making less updates is that coins that were issued within the last uncovered period can not be accepted by the merchant.
- When a user tries to spend a coin to a merchant who does not accept coins from the last k minutes, the only information the user reveals is that his coin was not withdrawn during the last k minutes.

We believe that such a system might offer a flexible and practical solution to anonymous off-line cash. We call the system off-line because if a merchant chooses to be updated only once a day he can certainly do so. The users can also spend their money without involving the bank. They suffer however from the disadvantage that at least at the beginning they need to be updated often. This can be improved if more recent roots are kept by the merchant.

3.3 On the Security of the System

A crucial property we need is that the integrity of the published root of the tree is maintained. This can be achieved by publishing the root in the New York Times, or mirroring it in many different places, as was already pointed out in the original paper of Merkle.

Besides the authenticated data the bank keeps all sorts of data structures, including, e.g., the tree itself, a data structure carrying the balance of each account in the system, and more. It is clear that the bank can change these data structures, and in fact this can be done even in the banking system of today, where a bank can technically take all the money from one's account. Such accounting problems are well understood, and they are quite successfully dealt with in the banking industry. We do not deal with them in our system.

In an off-line system a merchant needs to verify membership to L reliably, i.e. he needs the correct roots. Criminals might establish a site that claims to contain the necessary authenticated data, and try to trick merchants into accepting a forged root. There are many different ways to address this problem, and furthermore such an attack is detected by the bank as soon as a forged coin is deposited. The bank can take a variety of steps immediately like shutting down the false source of the root, raising an alert or redistributing the correct root. The merchant can also set his own policy how to verify the authenticity of the root, and thereby manage his risk.

4 System

We first describe our model:

The participants: Users, merchants, a bank and the auditor.

<u>Infrastructure</u>: We assume there is an authenticated way for the bank to distribute the roots of its trees of issued coins.

<u>Time</u>: We assume that there are consecutive time frames denoted T_1, T_2, \ldots We call the basic time frame a 'minute'. Two minutes are grouped together to an hour, two hours to a day and so forth.

Computing Power: All participants are probabilistic polynomial time players. <u>Trust Model</u>: The network and the distribution channels are reliable and anonymous. Users and merchants trust the bank not to steal their money ². The auditor does not trust the bank not to issue unreported money.

System Events: We focus on the following system events:

- A user opens an account.
- A user withdraws money at the bank.
- The bank updates and broadcasts the roots of the forest.
- A user pays a coin to a merchant.
- A merchant deposits a received coin at the bank.

 $^{^2}$ By introducing receipts this trust requirement can be minimized using well known techniques. In this work we focus on the auditability property.

- The bank invalidates funds that have been withdrawn before.

We have the following requirements for our system:

<u>Unforgeability</u>: It is infeasible for any coalition of participants in the system excluding the bank to create an amount of payments accepted by the bank that exceeds the amount of withdrawn coins.

<u>Auditability</u>: There is a file, accessible by the auditor, that is supposed to describe all the events that have occurred in the system. In particular all withdrawals are supposed to be reported there. A withdrawal record should at least report who withdrew the money. We say c is a coin if it can be deposited, or if it was already successfully deposited.

Definition 1. A system is auditable if there is a one to one correspondence between all coins c and the withdrawal records.

We stress that we do not require that the one to one correspondence is known to the auditor or anyone else. In fact, in a truly anonymous system it is not. All we require is that such a correspondence exists. If the system is auditable, we also say the system does not admit any unreported money.

<u>Non-Rigidness</u> We say a system is non-rigid if any coin that can be accepted as a valid payment by the deposit protocol (and it does not matter whether it was withdrawn in a standard or non-standard transaction) can be invalidated.

<u>Unconditional Payer Anonymity</u>: A payer has unconditional anonymity, if transcripts of withdrawals are statistically uncorrelated to transcripts of payments and deposits.

We stress that at withdrawal time the user has to identify himself to the bank, and the bank might record the withdrawn string z along with the identity of its owner. Yet, as transcripts of withdrawals are statistically uncorrelated to transcripts of payments and deposits, this does not give the bank any information on how or to whom a withdrawn coin is spent.

5 Tools

Definition 2. We say a function $f: A \times B \to C$ is one-way, if the probability a polynomial time machine given a random $c \in C$ can find (x,r) s.t. f(x,r) = c is negligible. We say a function $f: A \times B \to C$ is collision resistant, if the probability a polynomial time machine can find $(x,r) \neq (x',r')$ s.t. f(x,r) = f(x',r') is negligible.

Definition 3. Let G be a domain of size p. We say a function $g:[0..p-1] \times [0..p-1] \to G$ is concealing if for any [0..p-1] the distribution g(x,[0..p-1]) obtained by picking $r \in [0..p-1]$ at random and computing g(x,r) is the uniform distribution over G.

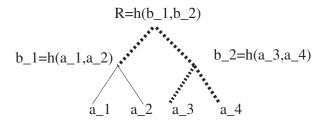


Fig. 1. A hash chain $((1,0); a_3; (a_4,b_1))$.

Assuming DLog is hard for certain groups of prime order G (see [9]) one-way, collision resistant and concealing functions exist and can be based on the representation problem [9]. More specifically, if G is a group of prime order p, for which DLOG is hard, and g_1, g_2 are chosen at random (so almost always they are two distinct generators of G), then $g: [0..p-1] \times [0..p-1] \to G$ defined by $g(x,y) = g_1^x g_2^y$ has these properties (see [9] Proposition 7 and Corollary 8, for details).

5.1 Hash Chains and Hash Trees

Hash Chains. A hash chain of length 1 to a root R is a triplet $(i_1; x; y)$ s.t. $f^{(i_1)}(x, y) = R$, where $f^{(0)}(x, y) = h(x, y)$ and $f^{(1)}(x, y) = h(y, x)$.

A chain of length d > 1 to a root R is a triplet $((i_1, \ldots, i_d); x; (y_1, \ldots, y_d))$ s.t. $((i_1, \ldots, i_{d-1}); f^{(i_d)}(x, y_d); (y_1, \ldots, y_{d-1}))$ is a hash chain of length d-1. We also say that the hash chain *starts* with the value x and leads to the root R. See Figure 1.

Hash Trees. For a given domain D, and a known hash function $h: D \times D \to D$ a hash tree (T, val) consists of a balanced binary tree T, with vertices V, together with a function $val: V \to D$ s.t. for any vertex v with two children v_1 and v_2 , $val(v) = h(val(v_1), val(v_2))$. The only operation that can be performed on a hash tree is UPDATE(leaf, w) where the leaf's value is changed to w and the values of the internal nodes from the leaf to the root are accordingly updated.

5.2 Non-interactive Zero Knowledge Arguments of Knowledge under the Random Oracle Assumption

We will frequently use perfect zero knowledge arguments of knowledge (ZKA) i.e., proofs that show that the prover knows a witness w to the predicate ϕ (i.e. $\phi(w) = True$). These proof are convincing if the prover is polynomially bounded, and the proofs statistically do not reveal extra information. The notion of a proof of knowledge is from [23,5]. Under the discrete log assumption any NP predicate has a perfect zero knowledge argument of knowledge ([11, 12, 22, 21], see also [29]

for zero knowledge arguments under weaker conditions and [4] for further details on ZKA's of knowledge).

We will need *non-interactive* perfect zero knowledge arguments (ZKA) of knowledge. We make the random oracle assumption [6] that has been commonly used in the design of electronic cash systems. Assuming the random oracle assumption, and using the techniques of Bellare and Rogaway [6], the zero knowledge argument of knowledge protocols can be made non-interactive (See [6] for the treatment of the similar case of zero-knowledge proofs).

6 An Anonymous Auditable Electronic Cash System

We now build an auditable, electronic cash system that allows users to have unconditional anonymity.

When Alice withdraws a coin, she chooses x and r (which she keeps secret) and sends z=g(x,r) to the Bank. x should be thought of as the serial number of the coin, r is a random number and g is concealing and collision resistant. The bank adds the coin z to the public list of coins, yet only a person who knows a pre-image (x,r) of z can use z for payment, and because g is one-way only Alice can use the coin z. To make the system anonymous for Alice when she wants to spend a coin z she proves with a zero knowledge argument that she knows a pre-image (x,r) of some z that appears in the list of coins, without actually specifying the value z. To prevent double–spending extra standard mechanisms are added to the system.

As it turns out the most expensive part of the protocol is the zero knowledge argument of membership where Alice has to prove she knows a coin z from the public list of coins that has certain properties. A straight forward implementation of this would require a proof that is polynomially long in the number of coins. A better, and theoretically efficient, protocol for testing membership in a list was suggested by Merkle [27] using hash trees, and our system builds upon this solution. We now give a more formal description of the protocol.

6.1 The Protocol

Setup: During system setup bank and auditor (and possibly users) choose jointly the following objects. F_q is a field of size q = poly(N) and N is an upper bound on the number of coins the bank can issue. G is a group of prime order p for which DLOG is hard, and $|G| \geq q^3$. Further an efficient 1-1 embedding $E: F_q^3 \to [0..p-1]$ is chosen (e.g., by using binary representations of the elements in F_q and concatenating them). $g: [0..p-1] \times [0..p-1] \to G$ is a one-way, collision resistant and concealing function.

D is a large domain, $|D| \ge |G|$, and $h: D \times D \to D$ a collision resistant hash function. An efficient 1-1 embedding $F: G \to D$ is chosen. The bank keeps a hash tree T over D with N leaves. This hash tree is gradually built. There is no need to initialize the tree.

Each merchant obtains a unique identifying identity, and we assume the existence of a random oracle $\mathcal{H}: TIME \times ID \to F_q$ that maps time and an id to a random element of F_q . We assume that each merchant can do at most one transaction per time unit (this time unit can be chosen to be very short). Alternatively the merchant has to add a serial number to each transaction occurring at the same time unit and is not allowed to use the same serial number twice.

Account Opening: When Alice opens an account, Alice has to identify herself to the bank, and the bank and Alice agree on a public identity $P_A \in F_q$ that uniquely identifies Alice.

<u>Withdrawal</u>: Alice authenticates herself to the bank. Alice picks $u_1 \in_R F_q$, $serial \in_R F_q$ and computes $u_2 = P_A - u_1 \in F_q$ and $x = (u_1, u_2, serial) \in F_q^3$, serial is the serial number of the coin and u_1, u_2 are used to encode Alice's identity. She also picks $r \in_R [0..p-1]$ and sends $z = F(g(E(x), r)) \in D$. to the bank. She gives the bank a non-interactive zero knowledge argument that she knows $u_1, u_2, serial$ and r s.t. $z = F(g(E(u_1, u_2, serial), r))$ and $u_1 + u_2 = P_A$, i.e, that the coin is well formed. The bank also makes sure that the coin z has not been withdrawn before. See Figure 2.

Fig. 2. Withdrawal

Alice Bank

$$u_1 \in_R F_q$$
, $u_2 = P_A - u_1 \in F_q$
 $serial \in_R F_q, r \in_R [0..p - 1]$
 $z = F(g(E(u_1, u_2, serial), r))$

 P_A, z

Alice gives a non-interactive ZKA that z is well formed

In the zero knowledge argument Alice has to answer challenges. These challenges are determined in the non-interactive ZKA protocol using the random oracle \mathcal{H} .

The bank then subtracts one dollar from Alice's account, and updates one of the unused leaves in the tree T to the value z (along with the required changes to the values along the path from the leaf to the root). When the time frame ends the bank takes a snapshot of the tree T and creates a version. After creating the version the bank sends Alice the hash chain from z to the root of T, taken from the hash tree T. Alice checks that she was given a hash chain from z to the public root of the hash tree T.

<u>Updates</u>: Each minute a new minute tree is generated, and a version of it is taken at the end of the minute. When two minute versions exist, they are

combined together to an 'hour' tree, by hashing the two roots together. Similarly, if two hour trees exist, they are combined together to a day tree and so forth.

At the end of each hour,day, week etc. a broadcast message is sent to all users who withdrew a coin during that time period. The hour update, e.g., contains the values of the two minute roots that were hashed together to give the hour tree root. Merchants can follow their own updating policy.

Payment: Alice wants to make a payment with a coin

 $z = \overline{F(g(E(u_1, u_2, serial), r))}$ to a merchant M with identity m_{id} . The protocol starts with M sending Alice the set ROOTS of live roots he knows. A root is alive if it is the root of the tree of the last minute, hour,day etc. Alice then sends the merchant serial, time and the value $v = u_1 + cu_2$, where $1 \neq c = \mathcal{H}(time, m_{id})$. She then proves to the merchant with a non-interactive ZKA that she knows u_1, u_2, r, R and a hash chain $((i_1, \ldots, i_d); w; (y_1, \ldots, y_d))$ to R s.t.

```
-R \in ROOTS,

-w = F(g(E(u_1, u_2, serial), r)) \text{ and }

-v = u_1 + cu_2.
```

The merchant verifies the correctness of the non-interactive ZKA. See Figure 3.

Fig. 3. Payment

Alice Merchant $c = \mathcal{H}(time, m_{id}) \in F_q$ $v = u_1 + cu_2 \in F_q$ $time, v, serial \\ time, v, serial$

Alice gives non-interactive ZKA that she knows a valid hash chain with the above c, v, serial, ROOTS

Deposit: The merchant transfers the payment transcript to the bank. The bank checks that the merchant with identity m_{id} has not earlier deposited a payment transcript with this particular parameter time. The bank verifies that the challenges are correct (i.e., they are $\mathcal{H}(time, m_{id})$), that the set ROOTS in the payment transcript consists of valid roots, and that the non-interactive ZKA is correct. The bank then checks whether serial has already been spent before. If not the bank credits the merchant's account and records $serial \in F_q$ as being spent along with the values $c \in F_q$ and $v(=u_1 + cu_2) \in F_q$.

If serial has been spent before the bank knows two different linear equations $v_1 = u_1 + c_1u_2$ and $v_2 = u_1 + c_2u_2$. The bank solves the equations to obtain u_1 and u_2 , and $P = u_1 + u_2$. The bank then finds the user with the public identity P.

<u>Invalidation of funds</u>: The bank removes the coins that should be invalidated from the forest and recomputes the corresponding roots and the hash chains for each leaf in the forest. The bank distributes the updated snapshot of the forest and sends the updated hash chains for each of the withdrawn coins in the forest to the user who withdrew it.

6.2 Correctness

Theorem 1. Under the random oracle assumption, if DLOG is hard, and the assumptions we made in the description of our model in Section 4 the electronic cash system is statistically anonymous, unforgeable, auditable, non-rigid and detects double spenders. The system is also efficient, and the time required from all players at withdrawal, payment and deposit time is polynomial in the \log of the total number of coins N (i.e. the depth of the tree) and the security parameter of the system. Each user receives $\log N$ messages per withdrawal.

Proof. Unforgeability. We prove that any spent coin must appear as a leaf in T. To spend a coin Charlie needs to know a hash chain $((i_1, \ldots, i_d); w; (y_1, \ldots, y_d))$ to a root R of a valid tree T. Because h is collision resistant, and all the players are polynomial, they can not find two different chains, with the same (i_1, \ldots, i_d) leading to the same value. Hence, it follows that the hash chain Charlie knows is exactly the path in T going down from the root according to (i_1, \ldots, i_d) . In particular w is a leaf of T.

<u>Double Spending</u>: If Charlie spends a coin z then this coin must have been withdrawn before, as was shown in the proof of the unforgeability property, let's say by Alice. When Alice withdrew z she proved she knew $u_1, u_2, serial$ and r s.t. $z = F(g(E(u_1, u_2, serial), r))$ and $u_1 + u_2 = P_A$. When Charlie spends the coin he proves he knows a hash chain that starts at z, and $u'_1, u'_2, serial', r'$ s.t. $F(g(E(u_1, u'_2, serial'), r')) = z$. As g is collision resistant and E and F are 1-1, it follows that $u'_1 = u_1, u'_2 = u_2$ and serial' = serial. In particular, any time a coin z is spent, the serial number that is reported is always the same (the serial number chosen at withdrawal time). In particular double spent coins can be easily detected by observing the serial number.

When a coin is spent a challenge to $u'_1 + cu'_2$ is answered and answered correctly (we have a non-interactive zero knowledge argument of correctness). Furthermore, as we have seen above the u'_1, u'_2 are always the same and they are identical to the u_1, u_2 chosen at withdrawal time. By the random oracle assumption, the assumption that each merchant has a distinct ID and that he can only do one transaction per time unit, no challenge is repeated twice. Hence if the same coin is spent twice, we see the answers to different challenges and we can easily solve the system of two equations to discover u_1 and u_2 and consequently $u_1 + u_2 = P$ which is the identity of the double spender.

<u>Anonymity</u>: The non-interactive zero knowledge arguments do not reveal any information in an information theoretic sense about Alice's inputs beyond the fact that the coin is well formed (resp. the validity of the claim that is proved). We

would like to show that the information at withdrawal time (i.e. P and z) is statistically independent of the information at deposit time $(c, v = u_1 + cu_2, serial$ and ROOTS). We first observe that even given $u_1, u_2, c, serial$ and ROOTS the value $z = F(g(E(u_1, u_2, serial), r))$ is uniform, because g is concealing. We are left with P at withdrawal time and $c, v = u_1 + cu_2, ROOTS$ at deposit time. Since $u_2 = P - u_1$ we have $v = cP + (1 - c)u_1$. As u_1 is uniformly distributed, so is v. Also, c is a function of time and merchant id (which are public data) and is independent of the coin.

The only possible information leakage can result from the subset ROOTS the user uses. However, ROOTS covers all the time periods acceptable by the merchant. Hence the deposit does not reveal more information then merely the fact that some user paid a merchant M a coin that was proved to be valid according to the update policy of the merchant. Thus, the system is anonymous. Merchants who try to provide users with a manipulated list ROOTS can be detected by the user (e.g., if he knows a valid root not in the list that should be there) and will definitely be caught by the bank (or the auditor) when they check the payment transcripts.

<u>Auditability</u>: We have already shown that if a polynomial time player can spend a coin c then it must have appeared as a leaf in the tree. As all the leaves in the tree are different, this shows that a one to one correspondence between usable coins and leaves in the tree (and therefore withdrawals) exists.

<u>Non-Rigidness</u>: We have seen that a coin z is only usable for payment if z appears as a leaf in the tree. Furthermore, the bank and the auditor know who withdrew z. To invalidate z all the bank has to do is replace z with a NULL, and update the tree, the hash chains and the roots as described in the system event "invalidation of funds".

Efficiency:

Withdrawal, Payment, Deposit: The dominating complexity stems from the zero knowledge arguments. Each non-interactive ZKA of an NP statement takes resources polynomial in the length of the NP statement. The way we described the protocol, the user claims knowledge of a valid root, and a hash chain leading to that root. Thus, the NP statement is of length $O((\log(N) + |ROOTS|) \log(|D|)$, (A root among all possible roots is a claim of length $O(|ROOTS| \log(|D|))$, and a hash chain is a sequence of $O(\log(N))$ elements each of length $O(\log(|D|))$). Thus, altogether the statement size is $O(\log^2 N)$.

Updates and Invalidation: Whenever a minute, hour, day, year etc. ends, the bank has to broadcast two root values to all the users who withdrew money during that period. E.g., at the end of the year the bank has to update all customers who withdrew money during that year. Minute updates are done frequently but involve only few users, year updates happen only once a year but involve many users. Each user will ultimately receive $\log N$ update messages each containing two root values. We point out that the bank can avoid notifying the users about year (and all infrequent) updates by simply publishing the two roots in the New York Times or a similar medium. It should also be noted that once

a broadcasting technology becomes widely accessible, the complexity of updates is only a constant.

Invalidation takes O(N) applications of the hash function to update the tree, and messages should be sent to all users. Unlike regular updates this update involves the whole tree (with O(N) data items) and can not be done by broadcast. However, invalidation should not happen too often as mentioned above, and is mainly used to deal with extreme cases where the bank surrendered to black-mailing.

7 Some Comments

Other protocols for testing membership in a list exist. In particular, the protocol suggested by Benaloh and de Mare [7] using one-way accumulators, has the property that it produces hash chains of length 1. The later protocol, however, suffers from the disadvantage that the coalition of parties which generates the one-way accumulator h knows a trapdoor for h. Employing the protocol in our payment system will reduce the length of the zero knowledge proofs to a constant independent of the number of withdrawn coins. However a person who knows the trapdoor, can completely bypass the system's security and pay with counterfeited money. Thus, if an honest agency constructs h and then destroys the trapdoor the system is safe. But at least during system setup the payment system is still vulnerable to the bank robbery attack as the trapdoor may be compromised. We consider this a serious disadvantage 3 .

We leave open the question of designing a practically efficient system that is both anonymous and auditable. We believe that by designing an appropriate accumulator—like hash function and the efficient corresponding zero knowledge proofs this should be possible. A first step in this direction was taken in [35], where efficient accumulators without trapdoor are constructed.

Finally, we ask if there are conceptually different ways to design anonymous payment systems which have the auditability property, and how existing anonymous electronic payment systems can be transformed into auditable ones with minimal performance overhead.

Acknowledgments

We would like to thank Moni Naor and Moti Yung for stimulating conversations and Mike Just for a having pointed out a reference to us.

³ Benaloh and de Mare leave open whether one-way accumulators can be built without using a trapdoor. Nyberg [30] builds such a system, with an accumulated hash of length $N \log(N)$, where N is the number of items to be hashed. The large hash length makes this protocol completely irrelevant to our case.

References

- 1. FATF-VII report on money laundering typologies. Financial Crimes Enforcement Network Publications, August 1996. http://www.treas.gov/fincen/pubs.html.
- Cyberpayments and money laundering. RAND, 1998. http://www.rand.org/publications/MR/MR965/MR965.pdf/.
- 3. D. Bayer, S. Haber, and W. S. Stornetta. Improving the efficiency and reliability of digital time-stamping. In R. M. Capocelli et al., editor, *Sequences II: Methods in Communication, Security and Computer Science*, pages 329–334. SV, New York, 1992.
- Bellare, Jakobsson, and Yung. Round-optimal zero-knowledge arguments based on any one-way function. In EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT, 1997.
- M. Bellare and O. Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, Advances in Cryptology—CRYPTO '92, volume 740 of Lecture Notes in Computer Science, pages 390–420. Springer-Verlag, 1993, 16–20 August 1992.
- Mihir Bellare and Phillip Rogaway. Random oracles are practical: A pardigm for designing efficient protocols. In Victoria Ashby, editor, 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, November 1993. ACM Press. also appeared as IBM RC 19619 (87000) 6/22/94.
- J. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. Lecture Notes in Computer Science, 765:274-??, 1994.
- 8. J. Benaloh and M. de Mare. Efficient broadcast time-stamping. Technical Report 1, Clarkson University Department of Mathematics and Computer Sciences. August, 1991.
- 9. S. Brands. An efficient off-line electronic cash system based on the representation problem. In 246. Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, December 31 1993. AA (Department of Algorithmics and Architecture), CS-R9323, URL=ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9323.ps.Z.
- S. Brands. Untraceable off-line cash in wallet with observers. In Douglas R. Stinson, editor, Crypto 93, volume 773 of LNCS, pages 302–318. SV, 1993.
- 11. Brassard and Crepeau. Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond. In FOCS: IEEE Symposium on Foundations of Computer Science (FOCS), 1986.
- G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences, 37:156–189, 1988.
- 13. E. Brickell, P. Gemmell, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, pages 457–466. Sandia National Labs, January 1995.
- 14. Report by the Committee on Payment, Settlement Systems, and the Group of Computer Experts of the central banks of the Group of Ten countries. Security of electronic money. Publication of the Bank for International Settlements, Basle, August 1996. http://www.bis.org/publ/index.htm.
- 15. J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. *Lecture Notes in Computer Science*, 1146:33–, 1996.
- J. Camenisch, J. Piveteau, and M. Stadler. Fair blind signatures. In L. Guilloy and J-J. Quisquater, editors, Euro Crypt 95, LNCS, pages 209–219. SV, 1995.

- 17. J. Camenisch, J. Piveteau, and M. Stadler. An efficient fair payment system. In Clifford Neuman, editor, 3rd ACM Conference on Computer and Communications Security, pages 88–94, New Delhi, India, March 1996. ACM Press.
- G. Davida, Y. Frankel, Y. Tsiounis, and Moti Yung. Anonymity control in E-cash systems. In Rafael Hirschfeld, editor, Financial Cryptography: First International Conference, FC '97, volume 1318 of Lecture Notes in Computer Science, pages 1–16, Anguilla, British West Indies, 24–28 February 1997. Springer-Verlag.
- 19. E. Fujisaki and T. Okamoto. Practical escrow cash system. Lecture Notes in Computer Science, 1189:33–??, 1997.
- 20. General Accounting Office (GAO). Private banking: Raul Salinas, Citibank, and alleged money laundering. GAO/OSI-99-1, December 1998. http://www.gao.gov/monthly.list/dec98/dec9811.htm.
- 21. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, July 1991.
- 22. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design (extended abstract). In A. M. Odlyzko, editor, Advances in Cryptology—CRYPTO '86, volume 263 of Lecture Notes in Computer Science, pages 171–185. Springer-Verlag, 1987, 11–15 August 1986.
- Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal on Computing, 18(1):186–208, February 1989.
- 24. Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. Journal of Cryptology, 3(2):99–111, 1991.
- 25. M. Jakobsson and J. Muller. Improved magic ink signatures using hints. In proceedings of Financial Cryptography '99. Forthcoming volume in Lecture Notes in Computer Science, 1999.
- M. Jakobsson and M. Yung. Revokable and versatile electronic mony. In Clifford Neuman, editor, 3rd ACM Conference on Computer and Communications Security, pages 76–87, New Delhi, India, March 1996. ACM Press.
- R. Merkle. Protocols for public key cryptosystems. In IEEE, editor, Proceedings of the 1980 Symposium on Security and Privacy, April 14–16, 1980 Oakland, California, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1980. IEEE Computer Society Press.
- 28. D. M'Raihi. Cost-effective payment schemes with privacy regulation. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology—ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 266–275, Kyongju, Korea, 3–7 November 1996. Springer-Verlag.
- M. Naor, R. Ostrovsky, Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11, 1998.
- K. Nyberg. Fast accumulated hashing. In Dieter Grollman, editor, Fast Software Encryption: Third International Workshop, volume 1039 of Lecture Notes in Computer Science, pages 83–87, Cambridge, UK, 21–23 February 1996. Springer-Verlag.
- 31. Report of the working party on electronic money of the Group of Ten countries. Electronic money consumer protection, law enforcement, supervisory and cross border issues. Publication of the Bank for International Settlements, Basle, April 1997. http://www.bis.org/publ/index.htm.
- 32. Basle Committee on Banking Supervision. Core principles for effective banking supervision. Publication of the Bank for International Settlements, Basle, September 1997. http://www.bis.org/publ/index.htm.

- 33. Basle Committee on Banking Supervision. Risk management for electronic banking and electronic money activities. Publication of the Bank for International Settlements, Basle, March 1998. http://www.bis.org/publ/index.htm.
- H. Peterson and G. Poupard. Efficient scalable fair cash with off-line extortion prevention. Lecture Notes in Computer Science, 1334:463-??, 1997.
- 35. T. Sander. Efficient accumulators without trapdoor. Manuscript, 1999.
- 36. T. Sander and A. Ta-Shma. Flow control: A new approach for anonymity control in electronic cash systems. In proceedings of Financial Cryptography '99. Forthcoming volume in Lecture Notes in Computer Science, 1999.
- 37. Daniel R. Simon. Anonymous communication and anonymous cash. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 61–73. Springer-Verlag, 18–22 August 1996.
- 38. S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers and Security*, 11(6):581–583, October 1992.

Oblivious Transfer with Adaptive Queries

Moni Naor[⋆] and Benny Pinkas^{⋆⋆}

Dept. of Computer Science and Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel
{naor,bennyp}@wisdom.weizmann.ac.il

Abstract. We provide protocols for the following two-party problem: One party, the sender, has N values and the other party, the receiver, would like to learn k of them, deciding which ones in an adaptive manner (i.e. the ith value may depend on the first i-1 values). The sender does not want the receiver to obtain more than k values. This is a variant of the well known Oblivious Transfer (OT) problem and has applications in protecting privacy in various settings.

We present efficient protocols for the problem that require an O(N) computation in the preprocessing stage and fixed computation (independent of k) for each new value the receiver obtains. The on-line computation involves roughly $\log N$ invocations of a 1-out-2 OT protocol. The protocols are based on a new primitive, $sum\ consistent\ synthesizers$.

1 Introduction

Oblivious Transfer (abbrev. OT) refers to several types of two-party protocols where at the beginning of the protocol one party, the Sender (or sometimes Bob or B), has an input and at the end of the protocol the other party, the receiver (or sometime Alice or A), learns some information about this input in a way that does not allow the sender Bob to figure out what she has learned. In particular, in 1-out-of-N OT (OT_1^N) protocols the sender's input consists of N strings X_1, X_2, \ldots, X_N ; the receiver can choose to get one element X_I and does not learn any information about any other string, and the sender does not learn anything about I.

Recently Naor and Pinkas [23] suggested efficient constructions of OT_1^N based on protocols for OT_1^2 . In particular the overhead of the initialization work of the sender is O(N) invocations of a pseudo-random function (which can be realized by a block cipher) and the transfer requires only $\log N$ invocations of OT_1^2 . This low overhead makes the OT_1^N protocol rather efficient even if N is very large (say, even if there are a billion elements for Alice to choose from).

Many applications might require the receiver to obliviously obtain several elements held by the sender. It is very inefficient to invoke independent runs

^{*} Research supported by grant no. 356/94 from the Israel Science Foundation administered by the Israeli Academy of Sciences.

^{**} Research supported by an Eshkol Fellowship of the Israeli Ministry of Science.

of a OT_1^N protocol because of the O(N) overhead of the initialization phase (remember, N might be very large). An efficient protocol for k-out-of-N OT (OT_k^N) was described in [23], with complexity which is O(N) for the initialization phase and O(k) OT's for the transfer phase (we omit small logarithmic factors). However, that protocol required the receiver to get all k elements simultaneously. That is to say that she had to decide in advance which k elements to get, and was unable to adaptively pick the elements to be transferred to her¹.

We present several efficient protocols for k successive (possibly adaptive) oblivious transfers, an operation which we denote as $\mathrm{OT}_{k\times 1}^N$. The sender has to perform a single initialization of his input, which requires O(N) work. Each transfer requires only about $\log N$ OT_1^2 's. In some of the protocols the parameter k does not affect the complexity, and the protocol can even be used for N successive transfers.

1.1 Motivation

 $\mathrm{OT}_{k\times 1}^N$ protocols are useful whenever the following three properties are required:

- A large database should be queried in an adaptive fashion.
- The privacy of the party which performs the queries should be preserved.
- The owner of the database does not want to reveal to the other party more than a minimal amount of information.

We describe below three applications of this type:

Oblivious search: Bob owns a database which Alice wants to search. Suppose first that the database is sorted and Alice is using binary search. The two parties can invoke a $\operatorname{OT}_{\log N \times 1}^N$ protocol to perform this search without revealing to Bob the element that Alice is searching for, and while limiting Alice's knowledge to $\log N$ elements. Alternatively, the data elements can be ordered by a two-level hash, using two hash functions. The first function maps data elements into bins and the second function further maps the elements that were mapped to the same bin (this is how perfect hash functions [15] are constructed). Our protocols can be used to let Alice obliviously determine whether an element is is in the table. It first computes by herself the bin to which the element should have been mapped, then performs an oblivious transfer to get the (second) hash function that is used in that bin, and then another oblivious transfer to check the final location into which the element should have been mapped.

¹ The protocol uses several random mappings of the data elements to cryptographic keys. It consists of several stages, and the receiver learns the ith mapping only after stage i-1 is complete. For every k data elements that the receiver wishes to learn the protocol guarantees that the probability (taken over the mappings) that she is able to learn another element, is small. However once she knows all the mappings she is able to ask for k data elements that enable her to learn more elements. If this protocol were used for adaptive OT the receiver would have been able to learn all the mappings after the first transfer phase.

Patent or legal databases: Suppose that Bob holds a patent database. He does not want to give the whole database to other parties, but is willing to let other people search the database using a World-Wide-Web interface. Alice has a bright idea which she wants to patent and as a first step she wants to conduct a search for related patents. She is afraid that if she conducts the search on Bob's database he might learn what she is interested in and might reveal her idea. Alice and Bob can use $\mathrm{OT}_{k\times 1}^N$ to enable Alice to search Bob's database without revealing her queries to him. This solution also applies to searches in legal databases such as Lexis-Nexis or Westlaw.

Medical data: Suppose that Bob holds a database of medical information. For proprietary or privacy reasons Bob does not want to reveal the whole database to other parties but he is willing to let them use it for their research. Alice wants to conduct a research about a certain disease and has a list of patients that have this disease. She wants to search Bob's database for records related to these patients, but cannot reveal their identities to Bob. Alice and Bob can use $\operatorname{OT}_{k \times 1}^N$ to enable Alice to gather the relevant information from Bob's database.

1.2 Protocol Structure

 $\mathrm{OT}_{k\times 1}^N$ protocols contain two phases, for initialization and for transfer.

The initialization phase is run by the sender (Bob) who owns the N data elements. Bob typically computes a commitment to each of the N data elements, with a total overhead of O(N). He then sends the commitments to the receiver (Alice).

The transfer phase is used to transmit a single data element to Alice. At the beginning of each transfer Alice has an input I, and her output at the end of the phase should be data element X_I . The transfer phase typically involves the invocation of several OT_1^m protocols, where m is small (either constant or \sqrt{N}). In these OT's Alice obtains keys which enable her to open the commitment to X_I (the protocol uses commitments rather than simple encryptions in order to prevent Bob from changing the data elements between invocations of transfers). An $\operatorname{OT}_{k\times 1}^N$ protocol supports up to k successive transfer phases.

1.3 Correctness and Security Definitions

The definition of correctness is simple: The sender's input is $X_1, X_2, ... X_N$. At each transfer phase j (where $1 \le j \le k$) the receiver's input is $1 \le I_j \le N$ which may depend on all the previous information she learned. At the end of this transfer phase the receiver should obtain X_{I_j} . Note that this implies that the sender essentially commits to his inputs at the beginning of the protocol and cannot change the X's between transfers².

² Our protocols enable the sender to prevent the receiver from obtaining certain data elements at certain times (of course, independently of the items which were previously transferred). It is possible to amend our protocols to ensure an "all or nothing"

The definition of security is separated into two issues: protecting the receiver and protecting the sender.

The Receiver's Security - Indistinguishability: Given that under normal operation the sender gets no output from the protocol the definition of the receiver's security in a $\operatorname{OT}_{k\times 1}^N$ protocol is rather simple: for any step $1\leq t\leq k$, for any previous items I_1,\ldots,I_{t-1} that the receiver has obtained in the first t-1 transfers, for any $1\leq I_t,I_t'\leq N$ and for any probabilistic polynomial time \mathcal{B}' executing the sender's part, the views that \mathcal{B}' sees in case the receiver tries to obtain X_{I_t} and in case the receiver tries to obtain $X_{I_t'}$ are computationally indistinguishable given $X_1,X_2,\ldots X_N$.

The Sender's Security - Comparison with Ideal Model: Here the issue is a bit trickier, since the receiver (or whatever machine is substituted for her part) gets some information and we want to say that the receiver does not get more or different information than she should. We make the comparison to the *ideal implementation*, using a trusted third party Charlie that gets the sender's input $X_1, X_2, \ldots X_N$ and the receiver's requests and gives the receiver the data elements she has requested. Our requirement is that for every probabilistic polynomial-time machine \mathcal{A}' substituting the receiver there exists a probabilistic polynomial-time machine \mathcal{A}'' that plays the receiver's role in the ideal model such that the outputs of \mathcal{A}' and \mathcal{A}'' are computationally indistinguishable. This implies that except for X_{I_1}, \ldots, X_{I_k} that the receiver has learned the rest of $X_1, X_2, \ldots X_N$ are semantically secure.

1.4 Previous Work

The notion of 1-out-2 OT (OT_1^2) was suggested by Even, Goldreich and Lempel [16] as a generalization of Rabin's "oblivious transfer" [26] (oblivious transfer was also developed independently by Wiesner in the 1970's, but not published till [28]). For an up-to-date and erudite discussion of OT see Goldreich [18]. 1-out-of-N Oblivious Transfer was introduced by Brassard, Crépeau and Robert [3,4] under the name ANDOS (all or nothing disclosure of secrets). They used information theoretic reductions to construct OT_1^N protocols from N applications of a OT_1^2 protocol (lately it was shown that such reductions must use at least N OT_1^2 's in order to preserve the information theoretic security [12]). Our work builds upon the simple OT_1^N and OT_k^N protocols of [23], which are based on efficient computationally secure reductions to OT_1^2 .

1.5 Comparison to Private Information Retrieval (PIR)

Private Information Retrieval (PIR) schemes [8] allow a user to access a database consisting of N elements $X_1, X_2, \ldots X_N$ and read any element she wishes without the owner learning which element was accessed. Compared to Oblivious Transfer

behavior or the sender, i.e. that in each transfer phase he either lets the receiver learn any data element she chooses, or quits the protocol. Note that in any two-party protocol it is impossible to prevent a party from quitting the protocol.

protocols, the emphasis in PIR is on the communication complexity which must be o(N). On the other hand, PIR schemes do not protect the owner of the database and do not prevent the user from learning more than a single element.

The first constructions of PIR schemes were based on using separate databases which do not communicate, but more recent constructions [20,5] use only a single database. More recently attention was given to the question of protecting the database as well, i.e. that the user will not learn more than a single data element. A PIR scheme that enjoys this property is called SPIR (for Symmetric PIR). In [17] an information theoretic transformation of any PIR scheme into a SPIR scheme was proposed at the cost of increasing the number of servers (and introducing the assumption of a separation between the servers). The OT_1^N protocols of [23] enable a straightforward and efficient transformation of any PIR scheme into a SPIR scheme without increasing the number of database servers. In [7] PIR schemes with more involved queries (such as keyword retrieval) are discussed.

The $\mathrm{OT}_{k\times 1}^N$ protocols that we introduce would enable even more efficient future transformations from PIR to SPIR. They can be used to transform a protocol for k adaptive invocations of PIR to a protocol for k adaptive invocations of SPIR (currently there are no adaptive PIR protocols, but when such a protocol is introduced it would be possible to immediately transform it to a SPIR protocol).

On a more practical level, we believe that it is preferable to use the computation efficient OT^N_1 and $\operatorname{OT}^N_{k\times 1}$ protocols rather than the communication efficient PIR protocols. The $\operatorname{OT}^N_{k\times 1}$ protocols require O(N) communication at the end of the initialization phase and before the transfer phases begin, and the communication overhead of the transfer phases is negligible. For many applications the communication in the initialization phase is not an issue, and can be done using devices such as DVD's, jaz drives, or a fast communication network. In contrast, single server PIR protocols [20,5] are very costly to implement since each transfer requires O(N) exponentiations (which must be done after the receiver sends her query).

2 Cryptographic Tools

The protocols use three cryptographic primitives, sum consistent synthesizers which are introduced in Section 2.1, 1-out-of-2 Oblivious Transfer, and commitments.

Protocols for **1-out-of-2 Oblivious Transfer** (OT₁²) can be constructed under a variety of assumptions (see e.g. [3,16,1]). Essentially every known suggestion of public-key cryptography allows also to implement OT, (but there is no general theorem that implies this state of affairs). OT can be based on the existence of trapdoor permutations, factoring, the Diffie-Hellman assumption (both the search and the decision problems, the latter yields more efficient constructions) and the hardness of finding short vectors in a lattice (the Ajtai-Dwork cryptosystem). On the other hand, given an OT protocol it is a simple matter

to implement secret-key exchange using it. Therefore from the work of Impagliazzo and Rudich [19] it follows that there is no black-box reduction of OT from one-way functions.

Commitment schemes are used to make sure that the sender does not change values between rounds. In a commitment scheme there is a commit phase which we assume to map a random key k and a value x to a string $commit_k(x)$, and a reveal phase which in our case would simply be revealing the key k which enables to compute x. The commitment should have the properties that given $commit_k(x)$ the value x is indistinguishable from random, and that it is infeasible to generate a commitment yielding two different x's. The commitment protocols we use are those of Chaum et al. [6] in Section 4, and of Naor [22] in Section 5.

2.1 Sum Consistent Synthesizers

Our constructions of $\mathrm{OT}_{k\times 1}^N$ protocols are based on committing to the data elements using pseudo-random synthesizers with a special property, which we call sum consistency. Each transfer phase reveals information which is sufficient to reveal just one data element, but cannot be used in conjunction with information from other transfer phases. Sum consistent synthesizers can be constructed based on the Decisional Diffie-Hellman assumption or based on a random oracle. We present in Section 4 a $\mathrm{OT}_{k\times 1}^N$ protocol which uses synthesizers based on the Decisional Diffie-Hellman assumption. In Section 5 we present a construction of a $\mathrm{OT}_{k\times 1}^N$ protocol based on any sum consistent synthesizer.

Pseudo-random Synthesizers: Pseudo-random synthesizers were introduced by Naor and Reingold in [24]. A pseudo-random synthesizer S is an efficiently computable function of ℓ variables x_1, \ldots, x_{ℓ} , that satisfies the following property: given polynomially-many uniformly distributed assignments to each of its input variables, the output of S on all the combinations of these inputs is pseudo-random. Consider for example a synthesizer S(x,y) with two inputs. Then for random sets of inputs $\langle x_1, \ldots, x_m \rangle, \langle y_1, \ldots, y_m \rangle$, the set $\{S(x_i, y_j) | 1 \leq i, j \leq m\}$ (which contains m^2 elements) is pseudo-random. That is to say that this set is indistinguishable from a truly random set³.

We use this property of synthesizers in order to encrypt the data elements. For example, the elements can be arranged in a square and a random key could be attached to every row and every column (say, key R_i to row i, and key C_j to column j). The element in position (i, j) can be committed to using the combined key $S(R_i, C_j)$. It is ensured that the values of any set of combined keys do not leak information about the values of other combined keys.

We require an additional property from the pseudo-random synthesizers that we use: they should have the same output for any two input vectors for which the sum of the input variables is the same. For example, for a two dimensional

³ This is a special property which does not hold for any pseudo-random generator *G*, since it involves inputs which are not independent.

synthesizer S this implies that for every x_1, y_1, x_2, y_2 that satisfy $x_1+y_1=x_2+y_2$ it holds that $S(x_1, y_1) = S(x_2, y_2)$. More formally, the requirement is as follows:

Definition 1 ((sum consistent synthesizer)). A function S (defined over ℓ inputs in a commutative group) is a sum consistent synthesizer if the following two conditions hold:

- S is a pseudo-random synthesizer.
- For every x_1, \ldots, x_ℓ , and every y_1, \ldots, y_ℓ that satisfy $\sum_{i=1}^{\ell} x_i = \sum_{i=1}^{\ell} y_i$, it holds that

$$S(x_1, x_2, \dots, x_\ell) = S(y_1, y_2, \dots, y_\ell)$$

The sum consistency property does not contradict the pseudo-randomness of the synthesizer. Suppose that S is a two dimensional sum consistent synthesizer, and let $\langle x_1, \ldots, x_m \rangle$ and $\langle y_1, \ldots, y_m \rangle$ be two random sets of inputs, whose size is polynomial in the security parameter of the synthesizer. Then there is an exponentially small probability that there is a pair $(x_i, y_j), (x'_i, y'_j)$ for which $x_i + y_j = x'_i + y'_j$. If ℓ is large enough then an ℓ dimensional synthesizer might contain such pairs with non-negligible probability. However then the range of possible ℓ -tuple inputs is exponentially large, and the probability of sampling such a pair is exponentially small.

Construction 1 (Random oracle based sum consistent synthesizer) Let RO be a random oracle. A sum consistent synthesizer can be realized as

$$S(x_1, x_2, \dots, x_\ell) = RO(x_1 + x_2 + \dots + x_\ell)$$

This simple construction means that (1) it is plausible to assume that such functions exist, and (2) suggests a heuristic approach for constructing such functions using a "complex" function (e.g. MD5). We prefer the number-theoretic construction that we present next, but on the downside it requires modular exponentiations which are more complicated to compute than common realizations of "complex" functions.

Another construction of sum consistent synthesizers is based on the synthesizers of [25] whose security relies on the Decisional Diffie-Hellman assumption (the DDH assumption is introduced and discussed in Section 4.1 below)⁴.

Construction 2 (DDH based sum consistent synthesizer) Let $\langle G_g, g \rangle$ be a group and a generator for which the Decisional Diffie-Hellman assumption holds. Let the input values x_1, \ldots, x_ℓ be elements in $\{1, \ldots, |G_g|\}$. A sum consistent synthesizer can be realized as

$$S(x_1, x_2, \dots, x_\ell) = g^{x_1 x_2 \cdots x_\ell}$$

⁴ In fact, in the usual representation it seems that these synthesizers have the same output for input vectors for which the *multiplication* of the input elements is equal. It is possible to look at a different representation of the inputs which results in the same outputs for *sum consistent* inputs. Both representations are sufficient for our purposes.

We use sum consistent synthesizers S in the following way which is depicted in Figure 1. Suppose that the elements are arranged as entries in a square and are committed to using $S(R_i, C_j)$, as described above. Then for each transfer phase Bob can choose a random value r, and let Alice obtain one of the values $\langle R_1 + r, R_2 + r, \dots, R_{\sqrt{N}} + r \rangle$, and one of the values $\langle C_1 - r, C_2 - r, \dots, C_{\sqrt{N}} - r \rangle$. Alice can compute $S(R_i + r, C_j - r) = S(R_i, C_j)$ and obtain the key that hides data element (i, j). We should also make sure that Alice is unable to combine the values she obtains in different transfer phases, and this requirement complicates the protocols a little.

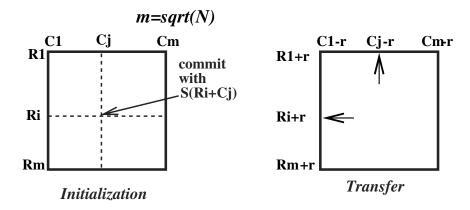


Fig. 1. The stages of the protocol.

3 The New $OT_{k\times 1}^N$ Protocols

We present two types of $\mathrm{OT}_{k\times 1}^N$ protocols of the above flavor, protocols whose security depend on the Decisional Diffie-Hellman assumption, and protocols which can be based on any sum consistent synthesizer. We start with two DDH based protocols. These protocols are somewhat simpler than the general construction, since the hardness of the discrete log problem prevents some attacks which are possible in the general case. The DDH based protocols can be used to transfer any number of elements. That is, they are good for $\mathrm{OT}_{k\times 1}^N$ with any k < N, and their efficiency does not depend on k. We then present a $\mathrm{OT}_{k\times 1}^N$ protocol based on any sum consistent synthesizer. This protocol is secure for at most k transfers, where k is a parameter which affects (logarithmically) the complexity of the protocol.

4 Protocols Based on the Decisional Diffie-Hellman Assumption

Following we present two protocols which are based on the Decisional Diffie-Hellman assumption. The protocols are very efficient, except that the basic operation they use is an exponentiation in a group in which the DDH assumption holds.

4.1 The Decisional Diffie-Hellman Assumption

The Decisional Diffie-Hellman assumption (DDH assumption) is used as the underlying security assumption of many cryptographic protocols (e.g. the Diffie-Hellman key agreement protocol [11], the ElGamal encryption scheme [14], the Naor-Reingold pseudo-random functions [25], and the Cramer-Shoup construction of a cryptosystem secure against chosen ciphertext attacks [10]). This assumption is very appealing since there is a worst-case average-case reduction which shows that the underlying decision problem is either very hard on the average case or very easy in the worst case [25,27].

The DDH assumption is thoroughly discussed in [2]. The assumption is about a cyclic group G_g and a generator g. Loosely speaking, it states that no efficient algorithm can distinguish between the two distributions $\langle g^a, g^b, g^{ab} \rangle$ and $\langle g^a, g^b, g^c \rangle$, where a, b, c are randomly chosen in $[1, |G_q|]$.

Our protocols essentially commit to the data elements using a key which is the output of the DDH based pseudo-random function or synthesizer of [25]. They are defined for a group G_g which is generated by a generator g, and for which the Decisional Diffie-Hellman assumption holds. It is very attractive to use in these protocols the non-interactive OT_1^2 protocols of Bellare and Micali [1]. The combination of these protocols with the proof techniques of [9] yields a very efficient OT_1^2 protocol which is based on the Decisional Diffie-Hellman assumption.

4.2 A Two-Dimensional Protocol

The following protocol arranges the elements in a two-dimensional structure of size $\sqrt{N} \times \sqrt{N}$. It uses $\mathrm{OT}_1^{\sqrt{N}}$ as a basic primitive (which can be efficiently implemented using [23]). In Section 4.3 we present a protocol which arranges the elements in a $\log N$ dimensional structure and uses OT_1^2 as its basic primitive.

Protocol 1 B's input is $X_1, X_2, ... X_N$, where $N = 2^{\ell}$. Rename these inputs as $\{x_{i,j} | 1 \leq i, j \leq \sqrt{N}\}$. In each invocation of the protocol the receiver A should learn a different element $X_{i,j}$.

1. Initialization:

(a) B prepares $2\sqrt{N}$ random keys

$$(R_1, R_2, \dots, R_{\sqrt{N}}) \ (C_1, C_2, \dots, C_{\sqrt{N}})$$

which are random integers in the range $1, \ldots, |G_g|$.

For every pair $1 \leq i, j \leq \sqrt{N}$, B prepares a commitment key $K_{i,j} = g^{R_i C_j}$, and a commitment $Y_{i,j}$ of $X_{i,j}$ using this key, $Y_{i,j} = commit_{K_{i,j}}(X_{i,j})$.

- (b) B sends to A the commitments $Y_{1,1}, \ldots, Y_{\sqrt{N},\sqrt{N}}$.
- 2. Transfer (this part takes place when A wants to learn $X_{i,j}$).

For each $X_{i,j}$ that A wants to learn the parties invoke the following protocol:

- (a) B chooses random elements r_R , r_C (r_R is used to randomize the row keys, and r_C is used to randomize the column keys).
- (b) A and B engage in a $OT_1^{\sqrt{N}}$ protocol for the values $\langle R_1 \cdot r_R, R_2 \cdot r_R, \ldots, R_{\sqrt{N}} \cdot r_R \rangle$. If A wants to learn $X_{i,j}$ she should pick $R_i \cdot r_R$.
- (c) A and B engage in a $OT_1^{\sqrt{N}}$ protocol for the values $\langle C_1 \cdot r_C, C_2 \cdot r_C, \dots, C_{\sqrt{N}} \cdot r_C \rangle$. If A wants to learn $X_{i,j}$ she should pick $C_j \cdot r_C$.
- (d) B sends to A the value $g^{1/(r_R r_C)}$.
- (e) A reconstructs $K_{i,j}$ as $K_{i,j} = (g^{1/(r_R r_C)})^{(R_i r_R) \cdot (C_j r_C)}$, and uses it to open the commitment $Y_{i,j}$ and reveal $X_{i,j}$.

The receiver can clearly obtain any value she wishes to receive in the above protocol (the sum consistency ensures that she reconstructs the same key that was used by B for the commitment).

As for the complexity of the protocol, the initialization phase requires B to compute all N commitment keys, i.e. to compute N exponentiations (see in protocol 2 a discussion on how to efficiently implement these exponentiations by utilizing the structure of the exponents). Each transfer phase requires two invocations of an $\mathrm{OT}_1^{\sqrt{N}}$ protocol (which each requires $O(\sqrt{N})$ initialization work by B).

The privacy of A is guaranteed by the security of the $OT_1^{\sqrt{N}}$ protocols which do not disclose to B information about A's choices. The use of commitments ensures that B cannot change the X_i 's between transfers.

The security of B is guaranteed by the Decisional Diffie-Hellman assumption. To show this we compare A to a party A' who instead of running the transfer phases simply asks and receives the keys for k commitments. We prove that A does not gain more information than A'. To complete the proof of security it is required to simulate A' and show that she does not learn more than k committed values. We present a theorem which states the security property and a sketch of its proof. The formal proof of this property turns out to be rather subtle since it involves the problem of selective decommitment which is described below.

Theorem 1 In k invocations of the above protocol A does not learn more information than a party A' which can adaptively ask and obtain the commitment keys of k elements.

Proof sketch: The security of the $\mathrm{OT}_1^{\sqrt{N}}$ protocol ensures that in each invocation of the transfer phase A can only learn a triplet of the following form $V_1 = (g^{1/r_1r_2}, R_ir_1, C_jr_2)$, where r_1, r_2 were chosen at random by B. This is equivalent to A learning a triplet $V_2 = (g^{R_iC_j/r_1r_2}, r_1, r_2)$. A can easily compute from this information the following tuple $V_3 = (g^{R_iC_j}, g^{R_iC_j/r_1r_2}, r_1, r_2)$ which

of course does not contain more information than the key $g^{R_iC_j}$ alone (the key enables A to generate tuples in the same distribution as that of V_3). The pseudorandomness of the output of the synthesizer ensures that A does not gain from k such keys more information about other keys than is available to A' which simply asks and receives k keys. It is left to prove that A does not learn anything from the commitments that she is given. This proof involves the selective decommitment problem, which we describe below.

The Selective Decommitment Problem and Its Resolution: Consider the following seemingly benign problem. A party A receives N commitments to N values. She then chooses k of the commitments, receives their keys, and opens them. It seems obvious that A does not gain information on the unopened values. A should therefore be simulated in the ideal model by a party who can ask and get k of the committed values and sees nothing else. Although there does not seem to be any obvious way for A to take advantage of the fact that she sees the commitments before asking for the keys, it is not known how to prove that this is indeed the case. The problem is that it is hard to simulate the operation of Asince it is unknown at the time of generating the commitments which of them she would ask to open. The number of the possible subsets of k commitments whose keys A might request is exponential. Note that it is possible to prove that A does not learn information about any other single element (or about a constant number of elements) since the simulation can with high probability "guess" the identity of this element. The selective decommitment problem is discussed in [13].

To enable the simulation it should be possible to open in the simulation any commitment to any value. Luckily, in the scenario of the $\mathrm{OT}_{k\times 1}^N$ there is an easy way to enable this. We describe below the solution for the DDH based protocol, which uses the trapdoor commitments of Chaum et al [6]. The solution for the protocols which use any sum consistent synthesizer is more complex and uses the commitments of Naor [22]. We do not describe it here.

In the case of DDH based $OT_{k\times 1}^N$ the protocol should be amended as follows (but for the sake of clarity we do not describe these modifications in the body of the protocols in the paper): (1) In the beginning of the protocol A should send to B two values $g_1, g_2 \in G$ and prove (in zero-knowledge) that she knows the discrete log of g_2 to the base g_1 . (In the simulation we would extract $\log_{g_1} g_2$ of the g_1, g_2 that would be used there). (2) B would use trapdoor commitments of the form $g_1^a g_2^b$ (which can be opened in an arbitrary way in the simulation, where $\log_{g_1} g_2$ is known). B commits to the value X_I in the following way, suggested in [6]: (i) chooses a random Y_I and computes $C_I = g_1^{X_I} g_2^{Y_I}$; (ii) takes the output of the synthesizer and uses it as a key to encrypt (X_I, Y_I) by xoring it, and sends these two results to A.

When the output of the synthesizer is computed it can be used to compute X_I and the commitment can be used to verify the result. In the simulation it is possible given X_I to find a Y_I which would be consistent with it, and give an output of the synthesizer which "decrypts" these values.

4.3 A Protocol Using OT₁²:

The following protocol uses simple OT_1^2 in a straightforward manner.

Protocol 2 B's input is $X_1, X_2, ... X_N$, where $N = 2^{\ell}$. In each invocation of the protocol the receiver A would like to learn a different element X_I .

1. Initialization:

(a) B prepares ℓ random pairs of keys

$$(a_1^0, a_1^1), (a_2^0, a_2^1), \dots (a_\ell^0, a_\ell^1)$$

where for all $1 \leq j \leq \ell$ and $b \in \{0,1\}$ each a_j^b is a random integer⁵ in the range $1,\ldots,|G_q|$.

For all $1 \leq I \leq N$ let $\langle i_1, i_2, \ldots i_\ell \rangle$ be the bits of I. B prepares a commitment key $K_I = g^{\prod_{j=1}^{\ell} a_j^{i_j}}$, and a commitment Y_I of X_I using this key, $Y_I = commit_{K_I}(X_I)$.

- (b) B sends to A the commitments $Y_1, Y_2, ... Y_N$.
- 2. **Transfer:** For each X_I that A wants to learn the parties invoke the following protocol:
 - (a) B chooses random elements r_1, \ldots, r_ℓ . Element r_i will be used to randomize the keys of the *i*'th coordinate.
 - (b) For each $1 \leq j \leq \ell$, A and B engage in a OT_1^2 protocol for the values $\langle a_j^0 r_j, a_j^1 r_j \rangle$. If A wants to learn X_I she should pick $a_j^{ij} r_j$.
 - (c) B sends to A the value $q^{1/r_1r_2\cdots r_\ell}$.
 - (d) A reconstructs K_I as $K_I = (g^{1/(r_1 r_2 \cdots r_\ell)})^{(a_1^{i_1} r_1) \cdots (a_\ell^{i_\ell} r_\ell)}$, and uses it to open the commitment Y_I and reveal X_I .

The receiver can clearly obtain any value she wishes to receive in the above transfer protocol.

The initialization phase requires B to compute all N commitment keys. This can be done with exactly N exponentiations if the order in which the keys are computed follows a Gray code (i.e. the Hamming distance between each two consecutive words is 1). The computation can be further improved by using efficient techniques for raising the same number to many powers, or for raising many numbers to the same exponent (see [21] for a survey of such techniques). It is an interesting problem to find a way to utilize the special structure of the exponents (being the multiplications of all the subsets of ℓ elements) to compute the $N=2^{\ell}$ commitment keys more efficiently.

The transfer part of the protocol requires $\ell = \log N$ invocations of an OT_1^2 protocol. In addition A and B should each compute a single exponentiation.

The privacy of A is guaranteed by the privacy of the OT_1^2 protocols. A is also ensured by the security properties of the commitments that B cannot change the values of the X_I 's between transfers.

The security of B is guaranteed by the Decisional Diffie-Hellman assumption, and is proven in a similar way to the security of protocol 1.

⁵ Note that B can set every a_j^0 to be equal to 1 without affecting the security of the system. This results in a reduction in the size of the keys that B needs to keep.

5 Protocols Based on Any Sum Consistent Synthesizer

We describe an *insecure* $\mathrm{OT}_{k\times 1}^N$ protocol which can be based on any sum consistent synthesizer, examine it, and transform it to a secure protocol.

5.1 An Insecure Protocol

The following protocol is **insecure**.

Protocol 3 (an insecure protocol)

B's input is $\{x_{i,j}|1 \leq i,j \leq \sqrt{N}\}$. Let S(x,y) be a sum consistent synthesizer with two inputs.

1. Initialization:

(a) B prepares $2\sqrt{N}$ random keys

$$(R_1, R_2, \dots, R_{\sqrt{N}})$$
 $(C_1, C_2, \dots, C_{\sqrt{N}})$

For every pair $1 \leq i, j \leq \sqrt{N}$, B prepares a commitment key $K_{i,j} = S(R_i, C_j)$ and uses the key to generate a commitment $Y_{i,j}$ of $X_{i,j}$, $Y_{i,j} = commit_{K_{i,j}}(X_{i,j})$.

(b) B sends to A the commitments $Y_{1,1}, \ldots, Y_{\sqrt{N},\sqrt{N}}$.

2. Transfer:

The parties invoke the following protocol for each $X_{i,j}$ that A wants to learn:

- (a) B chooses random elements r_R, r_C , such that $r_R + r_C = 0$ (r_R is used to randomize the row keys, and \underline{r}_C is used to randomize the column keys).
- (b) A and B engage in a $OT_1^{\sqrt{N}}$ protocol for the values $\langle R_1 + r_R, R_2 + r_R, \ldots, R_{\sqrt{N}} + r_R \rangle$. If A wants to learn $X_{i,j}$ she should pick $R_i + r_R$.
- (c) A and B engage in a $\mathrm{OT}_1^{\sqrt{N}}$ protocol for the values $\langle C_1 + r_C, C_2 + r_C, \ldots, C_{\sqrt{N}} + r_C \rangle$. If A wants to learn $X_{i,j}$ she should pick $C_j + r_C$.
- (d) A reconstructs $K_{i,j}$ as $K_{i,j} = S(R_i + r_R, C_j + r_C)$, and uses it to open the commitment $Y_{i,j}$ and reveal $X_{i,j}$.

The security problem: The above protocol enables A to learn any value she wishes and protects her privacy. However the protocol is insecure for B because A can combine information she learns in different transfers, and use linear relations between the keys to learn more keys than she is entitled to. For example she can use the relation $(R_i + C_j) + (R_{i'} + C_{j'}) = (R_{i'} + C_j) + (R_i + C_{j'})$. She can thus ask to learn the keys that generate $K_{i,j}, K_{i',j}$, and $K_{i,j'}$ and use the information she receives to illegally compute the key $K_{i',j'}$.

5.2 Fixing the Protocol

In order to transform the above protocol to be secure we use the following construction of a set of matrices. It is used to ensure the non-linearity of the information that A learns.

Construction 3 (k-out-of-N relation free matrices).

- Let M_1, \ldots, M_t be t matrices of size $\sqrt{N} \times \sqrt{N}$, each containing a permutation of all the elements $1, \ldots, N$.
- Consider a collection of N vectors, which each have $2t\sqrt{N}$ coordinates corresponding to the rows and columns of each of the matrices. Denote the coordinates as $\{(i,j,k) | 1 \le i \le t, 1 \le j \le 2, 1 \le i \le \sqrt{N}\}$ (i.e. i represents the matrix, j is either a row or a column, and k is the row (column) index).
- For each element $1 \le x \le N$ define a vector v_x . Denote the row and column of x in matrix i as $R_i(x), C_i(x)$. The coordinates in v_x that correspond to the locations of x in the matrices are set to 1. I.e. the 2t coordinates $(i, 1, R_i(x))$ and $(i, 2, C_i(x))$ are 1 and all other coordinates are 0.
- The t matrices are k-out-of-N relation free if the vectors corresponding to any k+1 elements are linearly independent.

For simplicity assume that the non-linearity is defined over the field GF(2). The following lemma suggests a random construction of a k-out-N relation free set.

Lemma 1 A random mapping of N elements to t matrices, where

$$t \ge \frac{\log(N/(k+1))}{\log(\sqrt{N}/(k+1))},$$

is with high probability k-out-of-n relation free.

Proof: The vectors contain $2t\sqrt{N}$ coordinates. Call the coordinates that correspond to the row (or column) keys of a certain matrix a *region*. The vectors contain 2t regions, each with \sqrt{N} coordinates. Each vector has in each region a single coordinate with a '1' value.

Consider a set of k+1 linearly dependent vectors. Then each coordinate either has no vectors in which it is set to 1, or it is set to 1 in at least two vectors. Therefore in each region the 1 values of all k+1 vectors are concentrated in at most (k+1)/2 coordinates.

Since the mapping to matrices locations is random the probability that this property holds for a single region is at most $(\frac{k+1}{2\sqrt{N}})^{(k+1)/2}$. The probability that it holds for all regions is therefore bounded by $(\frac{k+1}{2\sqrt{N}})^{(k+1)t}$. We apply the probabilistic method and require this probability to be smaller than the inverse of the number of subsets, $1/\binom{N}{k+1} \approx (\frac{k+1}{eN})^{k+1}$. This holds for $t \ge (\log(N/(k+1))/\log(\sqrt{N}/(k+1))$.

Note that this randomized construction is good for every $k = o(\sqrt{N})$. In particular, if $k < N^{1/4}$ then t = 3 satisfies the condition of the theorem, and for $k < N^{1/3}$ it suffices to use t = 4. It should be interesting to come up with an explicit construction of k-out-n relation free matrices (the problem seems to be related to the minimal weight of linear codes).

The transformation of protocol 3 is based on mapping the data elements to keys using a set of k-out-of-n relation free matrices. This ensures that the receiver

can learn at most k linear equations which correspond to k relevant synthesizer inputs. The full description of the secure protocol appears in Section 5.3. We first describe only the differences from protocol 3.

In the *initialization* phase B uses a k-out-of-N relation free construction of t matrices and maps the N elements to entries in the t matrices. Namely, the element whose index is x is mapped in each matrix to the entry which contains x. The sender publishes the mapping and makes it available to A. B chooses random keys for every row and column from each matrix (a total of $2t\sqrt{n}$ keys). The commitment key for each element is the output of a sum consistent synthesizer with 2t inputs, which are the keys corresponding to the rows and columns to which the element is mapped in each of the matrices.

In each transfer phase B chooses 2t random hiding elements r_i whose sum is 0. A and B run 2t $\mathrm{OT_1^{\sqrt{N}}}$ protocols, which let A learn all the relevant inputs to the synthesizer, each summed with the corresponding random element r_i . The sum of these values equals the sum of the synthesizer inputs that generated the key to the commitment, and so A is able to open it.

The following theorem states that this protocol is secure if enough matrices are used (fortunately, if k is not too close to \sqrt{N} very few matrices are needed).

Theorem 2 The above $OT_{k\times 1}^N$ protocol with a set of k-out-of-N relation free matrices is secure.

Proof sketch: The properties of the $\mathrm{OT}_1^{\sqrt{N}}$ protocol ensure A's privacy. It is required to prove that A cannot use the information she obtained in k invocations of the transfer protocol to learn more than k elements.

The protocol is run with t matrices which are k-out-of-n relation free. B uses in each transfer phase a new set of 2t random hiding elements, r_i , which sum to 0. Therefore A can learn in each transfer phase at most a single linear equation which does not involve the hiding elements. This equation is the sum of one key from each row and from each column.

In order to avoid the selective decommitment problem we present the proof assuming that B generated N encryptions of the data items, and not commitments. The proof for a protocol which uses commitments is more involved and uses the commitments of [22].

First assume that in each transfer phase A follows the protocol and learns an equation which corresponds to a key which was used to encrypt a data element. The relation freeness ensures that the k equations that A learns do not span any key which was used for encrypting another data element.

Note that for every synthesizer output that A obtains she also learns the sum of its inputs. In order to handle this in the simulation the encryptions there should be done with a different set of keys for each data element. I.e. instead of using $2t\sqrt{N}$ keys there would be N sets of 2t keys, so that the Ith set is used for encrypting to X_I . When A asks in the simulation for element X_I she receives the sum of the keys in the Ith set. The pseudo-randomness of the synthesizer ensures that she cannot distinguish this view from the real distribution.

Consider now the case in which A does not play by the rules and in some transfer phases asks to learn linear equations which do not correspond to any

of the encryption keys. Then in some later transfer phase she might learn an equation which, together with the previous equations she learned, reveals several keys. This might pose a problem in the simulation since it would be required to supply A with a single value which agrees with all the keys that she is supposed to learn in this phase. However observe that the j equations that A learns in the first j transfer phases span a subspace of dimension j of the N equations that were used for the encryptions. The value that A obtains in the jth phase of the simulation could be set to correspond to a new vector of this subspace which is learned by A in this phase . \square

5.3 A Protocol Based on Any Sum Consistent Synthesizer

The following protocol is a secure version of protocol 3.

Protocol 4 B's input contains the N elements $\{x_{i,j}|1 \leq i, j \leq \sqrt{N}\}$. B maps the inputs into a set of k-out-of-N relation free matrices, which contains t matrices. Let x_R^m and x_C^m denote the row and column into which x is mapped in matrix m.

Let $S(x_1, \ldots, x_{2t})$ be a sum consistent synthesizer with 2t inputs.

1. Initialization:

(a) B prepares $2t\sqrt{N}$ random keys

$$(R_1^1, R_2^1, \dots, R_{\sqrt{N}}^1) \ (C_1^1, C_2^1, \dots, C_{\sqrt{N}}^1), \dots, (R_1^t, R_2^t, \dots, R_{\sqrt{N}}^t) \ (C_1^t, C_2^t, \dots, C_{\sqrt{N}}^t)$$

For every pair $1 \le i, j \le \sqrt{N}$, B prepares a commitment key

$$K_{i,j} = S(R_{(x_{i,j})_R^1}^1, C_{(x_{i,j})_C^1}^1, \dots, R_{(x_{i,j})_R^t}^t, C_{(x_{i,j})_C^t}^t)$$

That is, the output of the synthesizer for the row and column keys that correspond to the locations of the input in each of the matrices. B prepares a commitment $Y_{i,j}$ of $X_{i,j}$ using this key, $Y_{i,j} = commit_{K_{i,j}}(X_{i,j})$.

- (b) B sends to A the commitments $Y_{1,1}, \ldots, Y_{\sqrt{N},\sqrt{N}}$.
- 2. **Transfer:** for each $X_{i,j}$ that A wants to learn the parties invoke the following protocol:
 - (a) B chooses random elements $r_R^1, r_C^1, \ldots, r_R^t, r_C^t$, such that their sum is 0. $(r_R^m \text{ is used to randomize the row keys of matrix } m, \text{ and } r_C^m \text{ is used to randomize the column keys of matrix } m).$
 - (b) For every matrix $1 \le m \le t$, A and B engage in the following protocols:
 - $A \text{ OT}_1^{\sqrt{N}} \text{ protocol for the values } \langle R_1^m + r_R^m, R_2^m + r_R^m, \dots, R_{\sqrt{N}}^m + r_R^m \rangle$. If $A \text{ wants to learn } X_{i,j} \text{ she should pick } R_{(x_{i,j})_r^m}^m + r_R^m$.
 - $A \text{ OT}_1^{\sqrt{N}}$ protocol for the values $\langle C_1^m + r_C^m, C_2^m + r_C^m, \dots, C_{\sqrt{N}}^m + r_C^m \rangle$. If A wants to learn $X_{i,j}$ she should pick $C^m(x_{i,j})_c^m + r_C^m$.
 - (c) A reconstructs $K_{i,j}$ as

$$K_{i,j} = S(R_{(x_{i,j})_R^1}^1 + r_R^1, C_{(x_{i,j})_C^1}^1 + r_C^1, \dots, R_{(x_{i,j})_R^t}^t + r_R^t, C_{(x_{i,j})_C^t}^t + r_C^t)$$

and uses it to open the commitment $Y_{i,j}$ and reveal $X_{i,j}$.

References

- M. Bellare and S. Micali, Non-interactive oblivious transfer and applications, Proc. Advances in Cryptology - Crypto '89, Springer-Verlag LNCS 435 (1990), 547-557.
- D. Boneh, The Decision Diffie-Hellman Problem, Proc. of the Third Algorithmic Number Theory Symposium, Springer-Verlag LNCS 1423 (1998) 48–63.
- G. Brassard, C. Crépeau and J.-M. Robert Information Theoretic Reduction Among Disclosure Problems, 27th Annual Symposium on Foundations of Computer Science, 1986, 168–173.
- G. Brassard, C. Crépeau and J.-M. Robert, All-or-Nothing Disclosure of Secrets, Proc. Advances in Cryptology - Crypto '86, Springr-Verlag LNCS 263 (1987), 234–238.
- C. Cachin, S. Micali and M. Stadler, Computationally Private Information Retrieval With Polylogarithmic Communication, Proc. Advances in Cryptology – Eurocrypt '99, Springr-Verlag LNCS 1592 (1999), 402–414.
- D. Chaum, E. van Heijst, and B. Pfitzmann, Cryptographically strong undeniable signatures, unconditionally secure for the signer, Proc. Advances in Cryptology – Crypto '91.
- B. Chor, N. Gilboa, and M. Naor, Private information retrieval by keywords, manuscript, 1998.
- 8. B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan, *Private Information Retrieval*, JACM 45 (1998), 965–981. Preliminary version appeared in Proc. 36th IEEE Symposium on Foundations of Computer Science, 1995.
- R. Cramer, I. Damgrd, B. Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols, Proc. Advances in Cryptology – Crypto '94, Springr-Verlag LNCS 839 (1994), 174–187.
- R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptove chosen ciphertext attacks, Proc. Advances in Cryptology – Crypto '98, Springr-Verlag LNCS 1462 (1998), 13–25.
- 11. W. Diffie and M. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory, vol. 22(6), 1976, 644–654.
- 12. Y. Dodis and S. Micali, Lower bounds for oblivious transfer reductions, Proc. Advances in Cryptology Eurocrypt '99, Springr-Verlag LNCS 1592 (1999), 42–54.
- C. Dwork, M. Naor, O. Reingold and L. Stockmeyer, Magic functions, manuscript, 1999.
- T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, Proc. Advances in Cryptology – Crypto '84, Springr-Verlag LNCS 196 (1985), 10–18.
- 15. M. L. Fredman, J. Komlos and R. Szemeredi, Storing a sparse table with O(1) worst case access time, JACM 31 (1984), 538–544.
- S. Even, O. Goldreich and A. Lempel, A Randomized Protocol for Signing Contracts, Communications of the ACM 28, 1985, 637–647.
- 17. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, *Protecting Data Privacy in Private Information Retrieval Schemes*, Proc. of the 30th ACM Symp. on the Theory of Computing, 1998.
- 18. O. Goldreich, Secure Multi-Party Computation (working draft) Version 1.1, 1998. Available at http://philby.ucsd.edu/books.html
- 19. R. Impagliazzo and S. Rudich, Limits on the Provable Consequences of One-Way Permutations, Proc. of the 20th ACM Symp. on the Theory of Computing, 1988.

- E. Kushilevitz and R. Ostrovsky, Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval, Proc. 38th IEEE Symp. on Foundations of Computer Science, 1997
- A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.
- M. Naor, Bit Commitment Using Pseudo-Randomness, Journal of Cryptology, vol. 4, 1991, 151–158.
- M. Naor and B. Pinkas, Oblivious Transfer and Polynomial Evaluation, Proc. 31th ACM Symp. on Theory of Computing, 1999, 245–254.
- M. Naor and O. Reingold, Synthesizers and their application to the parallel construction of pseudo-random functions, Proc. 36th IEEE Symp. on Foundations of Computer Science, 1995, 170–181.
- M. Naor and O. Reingold, , Number-Theoretic constructions of efficient pseudorandom functions, Proc. 38th IEEE Symp. on Foundations of Computer Science, 1997, 458–467.
- M. O. Rabin, How to exchange secrets by oblivious transfer, Tech. Memo TR-81, Aiken Computation Laboratory, 1981.
- M. Stadler, Publicly verifiable secret sharing, Proc. Advances in Cryptology Eurocrypt '96, Springr-Verlag LNCS 1070 (1996), 190–199.
- 28. S. Wiesner, Conjugate coding, SIGACT News 15, 1983, 78-88.

Compressing Cryptographic Resources

(Extended Abstract)

Niv Gilboa and Yuval Ishai

Computer Science Department Technion, Haifa 32000, Israel {gilboa,yuvali}@cs.technion.ac.il

Abstract. A private-key cryptosystem may be viewed as a means by which a trusted dealer privately conveys a large, shared pseudo-random object to a pair of players, using little communication. Alternatively, the messages distributed by the dealer may be viewed as a secure *compression* of a pair of large identical random pads (or random functions) into a shorter shared "key" or "seed".

We pose the question of extending this compression problem to more general correlation patterns among several players. Unlike the simple case of identical pads, where the main security concern is with respect to external eavesdroppers, in the case of general correlations participants also have to be protected from each other. That is, collusions of computationally-bounded players should gain no additional knowledge about the joint pads of the remaining players from the compressed messages they receive, other than what follows from the pads they generate and from knowing the joint distribution of all pads. While this ideal requirement is inherently impossible to meet using little communication, it turns out that it can be approximated to a satisfactory level, allowing to securely use such compressed correlated pads in a wide class of protocols. We propose a simple and modular replication-based approach for securely compressing any linear correlation pattern, using pseudo-random generators or pseudo-random functions in a black-box manner. Applications include amortizing the communication costs of private multi-party computation and proactive secret-sharing of large secrets.

1 Introduction

This work introduces and studies a natural generalization of the fundamental notions of pseudo-randomness (cf. [28,6,12,18]) to a useful and natural notion of privately correlated pseudo-randomness. Alternatively, our generalized notion may be viewed as providing a secure mechanism for compressing large correlated pads used as resources in cryptographic protocols.

1.1 Motivation

We consider a scenario in which a trusted dealer wishes to aid a group of players in performing some cryptographic task by supplying them with a resource of correlated messages. For instance, the dealer may allow a pair of players to communicate with unconditional secrecy by sending them a sufficiently long shared random one-time pad, or allow all players to form a secure multicast group by sending to all of them the same random pad.

As a second and slightly more sophisticated example, the dealer may aid the players to privately compute the modular sum of their inputs by granting them otherwise-random pads that add up to 0. Then, if each player adds his pad to his input and broadcasts the result, the broadcasted messages reveal no information about individual inputs other than their sum. Moreover, even collusions of players cannot learn from their view additional information about inputs of other players other than what follows from their inputs and the outcome of the computation (namely, the sum of all inputs).

The above solutions to the problems of secure communication and private addition possess several nice properties. First, they provide information theoretic security. Second, they are conceptually simple and computationally superefficient. Finally, they induce no on-line communication overhead: all extra communication is concentrated in an off-line distribution stage. Then, why aren't these solutions practical?

An obvious and prohibitive disadvantage of these solutions is their off-line communication cost.¹ Indeed, to maintain information theoretic security, each pad distributed by the dealer can only be used to mask a message of the same size; thus, if the players wish to encrypt long messages or perform many private additions, the dealer has to communicate sufficiently long pads. Even if we assume totally secure communication during the off-line distribution stage, the burden on the dealer would become unmanageable.

For the first example above, namely distributing a long shared random pad to two or more players, there is a standard way of reducing the off-line communication. The long shared pad may be replaced by a "compressed capsule" in the form of a seed to a pseudo-random generator or a key of a pseudo-random function. Such a seed or key may be "uncompressed" by the players holding it to obtain a large shared pseudo-random object which cannot be distinguished from a truly random object by external observers. However, for the second example of correlated pads which sum up to zero there seems to be no standard compression method.

Let us see why some simple approaches towards compressing the second kind of correlation fail. First, observe that if the same seed is sent to all players, it may be uncompressed to yield a distribution which is indistinguishable (by an external observers) from the one to be generated. However, such a solution totally fails to protect the privacy of players from each other: using this seed, each player will be able to recover the inputs of the other players from the broadcasted messages. A second attempt is to try to construct a pseudo-random generator such that given independently chosen seeds s_1, \ldots, s_{n-1} , it is always possible to find a last seed s_n such that the expansions of all seeds add up to

¹ This is a side from other important issues such as authentication and trust management, which should be handled by any private-key infrastructure and are independent of the concerns of this work.

(a long) zero. However, no such pseudo-random generator exists, as this would imply that its image forms a linear space.

In this work we pose the question of compressing more general correlation patterns and study the special case of *linear* correlations, where the pads form a random vector in a linear space.

How to Share a Zero. As paradoxical as this notion may sound, "secretly sharing a zero" among several players (i.e., using a secret-sharing scheme with the secret set to zero) is a common ingredient in cryptographic protocols. One example is the abovementioned application of privately computing the sum of ninputs. More generally, privately computing linear combinations of inputs is an essential subprotocol in general private multi-party protocols (e.g., [5,9]). But perhaps a better example is that of *proactive secret-sharing* and proactive multiparty computation (cf. [24,13,26]), where security has to be preserved over time against a mobile attack. In such applications, a shared representation of zero is periodically generated, either by an external dealer or by the players themselves, for the purpose of refreshing the distributed representation of some shared secret without changing its value. When the secret is very long, as may be the case when proactively sharing a large file, an expensive sharing of a "long zero" is required. One approach towards solving this problem (in the spirit of [17]) is to disperse an encryption of the large file among the players (without secrecy), and proactively share the short encryption key. However, a straightforward implementation of such a solution does not allow to retrieve or update a portion of the shared file without compromising the secrecy of the key. Moreover, some applications (e.g., protocols from [23]) crucially rely on the assumption that a non-encrypted representation of the entire file is shared using some simple linear secret-sharing scheme.

The above discussion motivates the problem of secretly sharing a "long zero" in a communication efficient way. While secret-sharing an *arbitrary* secret cannot use less communication than the length of the secret, it turns out that a shared long zero can be compressed.

1.2 Results

We focus on the case of *linear* correlations, where the pads form a random vector in a linear space. Moreover, while some of our results can be easily pushed to more adversarial scenarios, we adhere to a minimalistic model of an honest dealer and a passive adversary, keeping the problem as clean as possible.

We start by studying the information theoretic problem of private correlation from replication. This problem is motivated by its suitability for replacing perfect randomness with pseudo-randomness in a black-box manner. The setting is as follows. A dealer generates several independent random messages X_j , $1 \le j \le m$, and replicates them among n players (i.e., sends each message to a prescribed subset of the players). Then, each player P_i performs some local computation on the messages he received, resulting in some output value Y_i . We say that such a protocol privately generates a distribution D from replication, if: (1) the outputs (Y_1, \ldots, Y_n) are distributed according to D; and (2) each collusion of players

cannot learn from the messages X_j they receive any additional information about (Y_1, \ldots, Y_n) other than what follows from their outputs.

We obtain the following results:

- For the case of additively sharing a zero, we obtain an optimal scheme with $\binom{n}{2}$ messages, each sent to 2 players. This result is refined to privacy with respect to generalized adversary structures (in which privacy should only be maintained with respect to specified collusions), by reducing the problem to an interesting extremal graph-theoretic question. This refined result, in turn, has application to the construction of communicationand round-efficient (information theoretically) private protocols computing modular sum relative to general adversary structures.
- For any linear correlation, uniform over some linear space V, we obtain an optimal scheme whose complexity is proportional to the number of vectors in V with minimal support sets. For some linear spaces (such as the one corresponding to sharing a zero using Shamir's (n/2, n)-threshold secretsharing scheme [27]), the complexity of this solution will be exponential in the number of players.

We then address the possibility of replacing the random independent sources X_j by random independent seeds to a pseudo-random generator, thereby allowing to privately generate large correlated pseudo-pads using little communication. We show that while such a substitution cannot always preserve the security of an underlying information theoretically secure protocol, for a wide class of protocols meeting some mild technical requirement security is guaranteed to be preserved.

A major disadvantage of most of our solutions is their poor scalability; in some applications their complexity grows exponentially with the number of players.⁴ However, when the number of players is reasonably small, the advantages of our approach arguably outweigh this unfortunate byproduct of the replication paradigm. One such advantage is the ability to use any block- or stream-cipher in a black-box manner, which may result in orders of magnitudes of efficiency improvement over potential solutions based on specific number-theoretic constructions. Finally, we stress that the task of uncompressing correlated pads may be done off-line. This means that the on-line efficiency of protocols using our correlated pseudo-pads can be the same as the efficiency of protocols using perfect pads. For instance, suppose that a proactive secret-sharing protocol requires a large file to be shared using Shamir's secret-sharing scheme (for efficient on-line retrieval or updates of selected information). In each refreshment period correlated pseudo-pads are uncompressed and "added" to the current shares, and their seeds are erased. At the end of such a refreshment process, portions of

² Here and in the following, "optimal" should be interpreted as: "optimal under the replication paradigm".

The notion of private computation with respect to general adversary structures was first introduced and studied in [14].

⁴ Exceptions are resources used for private modular addition (relative to any adversary structure), or (t, n)-threshold secret-sharing of zero when either t or n - t are constant.

the shared file may be freely reconstructed or updated without any additional communication or computation overhead.

We finally note that while the main setting we consider when describing applications involves a trusted dealer, this is not an essential assumption. In fact, assuming that the players are honest but curious, the replication paradigm allows simulating the dealer by the players with no extra cost.

1.3 Related Work

The idea of treating correlated pads as useful cryptographic resources has been recently put forward in [3] under a specialized "commodity-based" model, and is implicit in many other works. However, to our knowledge the idea of *compressing* such general resources has never been treated.

Our paradigm of realizing private correlation via replication is related to the notion of replication-based secret-sharing schemes, which were introduced in [16] as a means of implementing secret-sharing relative to general access structures. Such schemes proceed by first additively sharing a secret, writing it as the sum (over a finite group) of otherwise random shares, and then distributing each additive share to some set of players associated with it. Replication-based secret-sharing schemes are implicitly used in [21,22] in the related context of sharing pseudo-random functions, and are used in [4] to realize private computation relative to general adversary structures and in [15] for obtaining efficient private information retrieval schemes. Some special cases of our constructions resemble (and in a sense generalize) the secret-sharing schemes used in these works; in general, however, both the semantics of the problem studied in this work and the replication-based combinatorial structures which underly our solutions are quite different. Our approach for compressing linear correlations may be viewed as a further, differently motivated, application of the replication technique.

Organization. The rest of this paper is organized as follows. Section 2 introduces some general notation. In Section 3 we define the information theoretic notion of private correlation from replication and study it for linear correlation patterns. In Section 4 we use the results of Section 3 to obtain privately correlated pseudorandom generators. Section 5 contains some concluding remarks. Finally, the appendices contain some definitions and a proof deferred from the main body of text.

2 Notation

By [n] we denote the set $\{1, 2, ..., n\}$. We use capital letters to denote random variables or probability distributions, and small letters to denote their instances. By $x \leftarrow X$ we denote a random choice of an instance x from the distribution X. All such random choices are made independently of each other. Additional context-specific notation will be defined in subsequent sections.

3 The Information Theoretic Setting: Private Correlation from Replication

In this section we study the information theoretic question of privately generating a joint distribution of correlated outputs from replicated *independent* random sources. This question is motivated by the possibility of replacing the independent random sources by seeds to a pseudo-random generator or a pseudo-random function, which will be explored in the next section.

A formal definition of this notion of *private correlation from replication* is given in the following subsection.

3.1 Definitions

Let X_1, X_2, \ldots, X_m be independent random variables, where each X_j is uniformly distributed over a finite domain \mathcal{X}_j , let $X = (X_1, X_2, \ldots, X_m)$ and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$. Let $\mathcal{Y}_1, \ldots, \mathcal{Y}_n$ be some finite sets, and $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n$. For $1 \leq i \leq n$, let f_i be some mapping from \mathcal{X} to \mathcal{Y}_i , and $S_i \subseteq [m]$ denote the set of arguments on which f_i depends (e.g., if $f_3(x_1, \ldots, x_m)$ depends on x_2 and x_5 and only on these arguments, then $S_3 = \{2, 5\}$). Finally, we let $f = (f_1, \ldots, f_n)$ (so that f maps from \mathcal{X} to \mathcal{Y}), and for $1 \leq i \leq n$ we define the random variable $Y_i = f_i(X)$, and $Y = (Y_1, \ldots, Y_n) = f(X)$. Notice that Y is a random variable distributed over \mathcal{Y} .

Before proceeding with the main definition, we briefly describe the correspondence between the above notation and the correlation from replication paradigm that motivates it. The random variables X_j are the independent random sources to be replicated among the n players. The function f (and its associated sets S_i) induces the following pad distribution protocol:

- The dealer generates $x \leftarrow X$ and sends to each player P_i , $1 \le i \le n$, all components x_j with $j \in S_i$.
- Each player P_i locally computes $y_i = f_i(x)$ based on the components of x he received,⁵ and outputs y_i as his random pad.

Notice that the function f concisely defines the entire protocol, including both the identity of messages sent to each player and the local computations of all players.

For any set $T \subseteq [n]$, we let $Y_T \stackrel{\text{def}}{=} (Y_i)_{i \in T}$ and $X_T \stackrel{\text{def}}{=} (X_j)_{j \in \cup_{i \in T} S_i}$. The random variable Y_T includes the joint outputs of the players P_i , $i \in T$, and X_T includes all messages they receive from the dealer.

We are now ready to define the notion of privately generating a given probability distribution from replicated independent sources.

Definition 1. Let D be a probability distribution over \mathcal{Y} , and $\mathcal{T} \subseteq 2^{[n]}$ be an adversary structure (specifying collusions of players to protect against). We say that a mapping f as above \mathcal{T} -privately generates D from replication of X (or \mathcal{T} -privately generates D for short) if the following two conditions hold:

⁵ Note that by the definition of the sets S_i , the x_j 's sent by the dealer are necessary and sufficient for this local computation to be possible.

Correctness. The output n-tuple Y(=f(X)) is distributed according to D. **Privacy.** For any collusion $T \in \mathcal{T}$, the random variable Y is independent of X_T given Y_T ; that is, for any $x \in \mathcal{X}$,

$$\Pr[Y = y \mid X_T = x_T, Y_T = y_T] = \Pr[Y = y \mid Y_T = y_T],$$

where y = f(x). Intuitively, the above privacy condition asserts that the messages X_T which members of T receive from the dealer give them no additional information about the joint output Y other than what follows from their outputs Y_T .

Finally, we say that f t-privately generates D if it T-privately generates D for the threshold structure $T = \{T \subseteq [n] : |T| \le t\}$, and that it privately generates D (or generates D with full privacy) if it n-privately generates D.

3.2 Additively Sharing a Zero

Before proceeding to study the general case of linear correlations, we start with the important special case of additively sharing a zero. Since the basic solution we provide for this special case will be generalized in the next subsection, we allow ourselves to omit some formal proofs and focus on the high-level intuition. Moreover, while in the general case we mainly address the question of constructing fully-private correlation generators, for the special case of additively sharing a zero we pay more attention to realizing privacy with respect to threshold and non-threshold adversary structures, and reduce the latter question to an interesting extremal graph-theoretic problem.

Fix a finite field F, ⁶ and define the distribution D_s as the uniform distribution over all n-tuples $(d_1, \ldots, d_n) \in F^n$ such that $\sum_{i=1}^n d_i = 0$. We first address the case of generating D_s with full privacy, and then proceed to study more general adversary structures.

To privately generate D_s from replication, consider the clique K_n whose n vertices are labeled by player indices, and assign an independent random source X_e to each edge $e = \{u, v\}$ in the graph. That is, let $m = \binom{n}{2}$, and label the m entries of X by edge names. Moreover, assume that the edges of the graph are arbitrarily directed (e.g., from the smaller numbered vertex to the larger one). Such a directed version of K_n , and similarly of any other graph, induces the following scheme for generating D_s : Each player P_i receives all sources X_e such that e is incident to its corresponding vertex, and outputs the difference

$$f_i(X) \stackrel{\text{\tiny def}}{=} \sum_{e \text{ enters vertex } i} X_e - \sum_{e \text{ exits vertex } i} X_e.$$

It can be easily verified that any such graph-based scheme satisfies the correctness requirement, since the contribution of each edge to the sum $\sum_{i=1}^{n} f_i$ is 0. An obvious *necessary* condition for the privacy requirement is that each

⁶ Results here and in the following section carry on to more general algebraic structures. For instance, here F may be replaced by any finite Abelian group.

collusion T of at most n-2 players misses at least one source X_e ; otherwise, the variables X_e held by such a collusion would completely determine the joint output Y, contradicting the privacy requirement. Indeed, this condition is met by the above clique-based scheme since each such player set T misses at least one edge. This necessary condition also shows that the complexity of the clique-based solution is optimal under our replication paradigm. This is argued as follows. For each collusion T containing exactly n-2 players, there must exist a source X_j (on which f depends) which T misses; moreover, X_j must be distributed to both players outside T, for otherwise the correctness requirement prevents f from depending on X_j . It follows that for each set T of size n-2 there exists a unique source X_j it misses, from which it follows that $m \geq \binom{n}{2}$.

In the case of full privacy, it turns out that the above necessary condition for privacy is also sufficient; however, as the following example shows, this is not the case for other adversary structures.

Example 1. Consider the replication scheme induced by a cycle of length n (whose edges are arbitrarily directed, as in the above case of a clique). Such a scheme satisfies the correctness requirement, and also satisfies the above necessary condition for privacy for |T| < n/2. Indeed, every set of less than n/2 vertices cannot cover all edges of the cycle. However, it is easy to see that there exist collusions of two players which can totally determine the output of a third player, thereby violating the privacy condition of Definition 1. For instance, in the scheme induced by the cycle $(P_1, P_2, P_3, \ldots, P_1)$, P_1 and P_3 can determine the output of P_2 since they cover all edges incident to him.

A necessary and sufficient condition for privacy should assert that for any collusion $T \in \mathcal{T}$, the dimension of the space of possible Y values given X_T is exactly n - |T| - 1, the dimension of Y given Y_T . Intuitively, each of these two dimensions corresponds to the number of "degrees of freedom" of the outputs of the remaining players given a view of members of T. The analysis in the next subsection (or the proof sketch of the following theorem) will show that the above condition is indeed satisfied by the clique-based scheme.

We now reduce the problem of \mathcal{T} -privately generating additive shares of zero to an extremal graph theoretic problem.

Theorem 1. Let $T \subseteq 2^{[n]}$ be an adversary structure, and let G(V, E) be a graph with V = [n]. Then, G induces a T-private generator of D_s if and only if for every $T \in \mathcal{T}$ the subgraph G - T, induced by removing all T-vertices from G, is connected.

Proof sketch. The proof follows from the fact that the rank of the incidence matrix⁸ of a graph equals its number of vertices minus its number of connected components (cf. [7, page 38]). This implies that if T does not disconnect G, then

Note that the above requirement on the subgraphs G-T is not monotone with respect to T (i.e., it is possible that $T' \subseteq T$ yet only the removal of T' disconnects G).

⁸ The incidence matrix of a graph G(V, E) is a $|V| \times |E|$ matrix M, such that for any edge $e = \{u, v\} \in E$ the e-th column of M contains 1 in its u-th entry, -1 in its v-th entry (or vice versa) and 0 elsewhere.

the dimension of the space of possible Y values given X_T is exactly n - |T| - 1 as required, and otherwise it is smaller.

The above connectivity requirement illuminates a qualitative difference between the type of combinatorial structures considered in our context and the ones required in the related context of replication-based secret-sharing [16,21]. Crudely speaking, the latter context only induces covering-type requirements on restricted unions of replication sets, which in our context are generally not sufficient (as demonstrated by Example 1).

For the special case of threshold structures, Theorem 1 suggests that a t-private generator of D_s can be based on any (t+1)-connected graph. For instance, when t is odd, a corresponding optimal graph may be obtained by connecting each vertex i to the (t+1)/2 vertices with (cyclically) consecutive indices.

We end this subsection by mentioning a further application of the above graph-theoretic characterization. By letting the players simulate the dealer (e.g., having each replicated source be picked and multicast by one player to which it is assigned), the above results can be used to obtain communication-efficient, perfectly-private two-round protocols for modular addition relative to general adversary structures. For the case of threshold structures, this can be compared with the (communication-optimal) private protocol of [10], which is only slightly more communication-efficient but requires many rounds of interaction.

3.3 Optimal Construction for Linear Correlations

We now move to the general case of linear correlations, where the joint output distribution D forms a uniformly random vector in some linear space over a finite field F. In the following we will stick to coding theory terminology: we will refer to the linear space as a linear code, and to a vector in this space as a codeword. To simplify notation, we only address the case where D is uniformly distributed over some length-n linear code $C \subseteq F^n$; that is, we focus on generating probability distributions in which each player P_i outputs the i-th coordinate of a uniformly random codeword $c \in C$. More general linear correlations, in which each player outputs several coordinates of a random codeword c, can be handled in a similar fashion.

Let D_C denote the uniform distribution over C. We will privately generate the distribution D_C using a linear generator of the form f(x) = Mx, where M is an n-row matrix over F. Note that in order to satisfy the correctness requirement alone, it suffices for the columns of M to generate the code C. This is not sufficient, however, for satisfying the privacy requirement. For instance, if M contains only nonzero entries, then each player views all of X and hence obtains full information on Y.

Our construction of a private generator for D_C will use a notion of minimal support codewords. For any codeword $c \in C$ we define the support set of c, denoted s(c), as the set of coordinates i such that $c_i \neq 0$. We say that a codeword c is of minimal support, if $c \neq 0$ and there is no $c' \in C$, $c' \neq 0$, such that $s(c') \subset s(c)$. It is straightforward to observe that if two codewords c, c' are both of minimal support and s(c) = s(c'), then one must be a multiple of the other (otherwise there exists a linear combination of c, c' whose support contradicts

their minimality). Hence the set of minimal support codewords in C can be naturally partitioned into equivalence classes, where two minimal support codewords are equivalent if one is a multiple of the other. Finally, we define Min(C) as a set of representatives of the above equivalence classes (e.g., Min(C) may include all minimal support codewords whose first nonzero coordinate is 1).

The following theorem gives our construction of a private linear correlation generator.

Theorem 2. Let C be a length-n linear code over F, and M be an $n \times m$ matrix whose m columns consist of all codewords in Min(C) (i.e., all representatives of minimal-support codewords in C). Then, the linear function $f_M: F^m \to F^n$ defined by $f_M(x) = Mx$ privately generates D_C from replication.

The proof of Theorem 2 uses the following sequence of lemmas. The (straightforward) proofs of the first two lemmas are omitted from this version.

Lemma 1. For any linear code C, span(Min(C)) = C.

Lemma 2. Let U^b denote the uniform distribution over F^b . Then, for any $a \times b$ matrix A over F, the random variable AU^b is uniformly distributed over the column-span of A.

It follows from Lemmas 1, 2 that Y = MX is uniformly distributed over C. Hence, the function f_M defined in Theorem 2 correctly generates D_C . It remains to show that it also satisfies the privacy requirement, with respect to any collusion $T \subseteq [n]$.

We Let x_T, y_T denote restrictions of instances x, y of the random variables X, Y, similarly to the notation X_T, Y_T defined in Subsection 3.1. (Note that this restriction has a different meaning when applied to an m-tuple x than when applied to an n-tuple y). We let $Y|x_T$ denote the conditional space induced by conditioning the random variable Y by the event $X_T = x_T$. Similarly, by $Y|y_t$ we denote the conditional space induced by conditioning the random variable Y by the event $Y_T = y_T$. Intuitively, $Y|x_T$ models the information learned by members of T from their view of X, and $Y|y_T$ models what they must infer from their outputs.

Lemma 3. For any $x \in F^m$, the conditional spaces $Y|x_T$ and $Y|y_T$ are identically distributed, where $y = f_M(x)$.

Proof sketch. Let $[Y|x_T]$ and $[Y|y_T]$ denote the supports of the corresponding spaces (i.e., the sets of $y \in F^n$ that have nonzero probability). First, writing the corresponding systems of linear equations, it is easy to observe that both spaces are *uniformly* distributed over some affine subspace of C. Hence, it suffices to show that the supports of the two spaces are equal, i.e. that $[Y|x_T] = [Y|y_T]$.

Let $C_{\overline{T}}$ denote the subspace of C consisting of all codewords c such that $c_T = 0$, and $M_{\overline{T}}$ denote the submatrix of M containing the rows with indices from $\overline{T} = [n] \setminus T$, restricted to columns from $C_{\overline{T}}$ (whose T-rows are all zero). That is, write M as

$$M = \begin{pmatrix} M_1 & 0 \\ M_2 & M_{\overline{T}} \end{pmatrix},$$

where the rows of M_1 are those labeled by members of T, and every column of M_1 includes at least one nonzero entry.

We now turn to show the equality $[Y|x_T] = [Y|y_T]$. The " \subseteq " inclusion follows immediately from the fact that x_T determines y_T (since y_T is the result of local deterministic computations performed on x_T). For the other inclusion, it suffices to show that $|[Y|y_T]| = |[Y|x_T]|$, or equivalently that $\dim[Y|y_T] = \dim[Y|x_T]$ (where $\dim V$ denotes the dimension of an affine space V). From the corresponding systems of linear equations one can deduce that:

$$\dim[Y|y_T] = \dim C_{\overline{T}} \tag{1}$$

$$\dim[Y|x_T] = \operatorname{rank} M_{\overline{T}}.$$
 (2)

It follows from the definitions of M and $M_{\overline{T}}$ that the columns of $M_{\overline{T}}$ (augmented with zeros as their T-entries) include all codewords in $\operatorname{Min}(C_{\overline{T}})$. By Lemma 1 these columns span $C_{\overline{T}}$, hence dim $C_{\overline{T}} = \operatorname{rank} M_{\overline{T}}$, and using equations (1),(2) above we obtain that $\dim[Y|y_T] = \dim[Y|x_T]$ as required.

It follows from Lemma 3 that the function f_M defined in Theorem 2 also satisfies the *privacy* requirement with respect to any collusion $T \subseteq [n]$, concluding the proof of Theorem 2.

The (fully-private) solution for additively sharing a zero, as described in the previous subsection, may be derived as a special case of Theorem 2. Another useful special case is a secret-sharing of zero using Shamir's scheme.

Corollary 1. Let D be the distribution induced by sharing 0 according to Shamir's secret-sharing scheme with parameters (t+1,n) over a finite field F (|F| > n). That is, D is generated by picking a random polynomial p over F of degree $\leq t$ and free coefficient 0, and outputting the n-tuple $(p(\alpha_1), \ldots, p(\alpha_n))$ (where $\alpha_1, \ldots, \alpha_n$ are some fixed distinct nonzero field elements). Then, D can be privately generated from replication with $m = \binom{n}{t-1}$, where each of the m independent field elements is sent to n-t+1 players.

Proof. It is not hard to observe that a distribution D as above is uniform over a linear code C such that Min(C) includes $\binom{n}{t-1}$ codewords, each of Hamming weight n-t+1. More generally, the same is true for any MDS code (cf. [19]) of length n and dimension t.

We end this section with a couple of remarks about the above construction.

Remarks:

- 1. On optimality. The construction of the correlation generator f_M in Theorem 2 can be shown to be optimal (under the correlation from replication paradigm), with respect to the communication complexity of the induced protocol. The proof of this fact may proceed similarly to the arguments made in the previous subsection for the special case of additively sharing a zero, and is omitted from this version.
- 2. On generalization to \mathcal{T} -privacy. The above solution can be made more efficient if the full privacy requirement is relaxed to \mathcal{T} -privacy. In fact, it is possible to derive from the proof of Theorem 2 a generalization of

Theorem 1, providing a necessary and sufficient condition on a subset of the columns of M so that the corresponding submatrix M' will \mathcal{T} -privately generate the distribution D_C . Specifically, the relaxed privacy requirement should assert that for every $T \in \mathcal{T}$, dim $C_{\overline{T}} = \operatorname{rank} M'_{\overline{T}}$, where $C_{\overline{T}}$ and $M'_{\overline{T}}$ are as defined in the proof of Lemma 3.

4 The Computational Setting: Privately Correlated Pseudo-Random Generators

In this section we exploit private correlation generators constructed in the previous section for implementing "privately-correlated pseudo-random generators". Intuitively, such generators should allow to securely substitute in applications long correlated perfectly-random pads by correlated pseudo-random pads, where these "pseudo-pads" are generated by locally uncompressing short seeds sent by the dealer. While the results shown in this section are not particularly strong, clean, nor elegant, we view them as an important demonstration of plausibility. We start by addressing the delicate issue of formally defining the desired computational notion.

A first (and technical) difficulty with incorporating the results of the previous section in a computational setting is the fact that for certain linear correlation patterns, the complexity of their generation from replication is exponential in n. (We stress though that for some useful correlations, such as additive shares of zero, this is not the case.) For simplicity, throughout the remainder of this section we fix a linear distribution D (which in particular determines the number of players n and the size of the field F to be constants), and treat the number of generated instances of D (corresponding to the length of the generated pads) as the major complexity parameter. While this "resolves" the above technical difficulty, a more inherent difficulty is discussed below.

The previous section described a procedure for privately generating a joint distribution D of n correlated pads from m independent and replicated random sources. Such a procedure may be defined using a more general terminology of private multi-party computation (cf. [8,1,20]) as follows. We consider a simple model consisting of n parties, connected to a trusted dealer via secure channels, and a passive, static T-adversary. The function f is a T-private generator for the distribution D (as defined in Definition 1) if it induces a perfectly T-private protocol computing the probabilistic n-output function that takes no inputs and outputs an n-tuple g distributed according to g Now, a natural way to define the notion of privately-correlated pseudo-random generators using general terminology of private multi-party computation is the following. Let ℓ denote the number of repetitions of the distribution p to be generated. The parameter ℓ , as well as a security parameter ℓ , will be given as inputs to all players. Moreover, to

To be consistent with the usual computationally-oriented definitions of private multiparty computation found in the literature, one also needs to add an extra requirement of efficient simulatability to Definition 1. However, for constant n (and arbitrarily large F) our solutions for linear correlations are easily seen to satisfy this extra requirement.

simplify definitions we assume that $\ell = p(\kappa)$ for some polynomial $p(\kappa) = \omega(k)$; that is, the length of the generated pads is some fixed superlinear polynomial of the security parameter. The distribution D on F^n induces a distribution D^ℓ on $(F^\ell)^n$, generated by ℓ independent repetitions of D. Notice that the information theoretic constructions of the previous section easily generalize from privately generating a distribution D to privately generating D^ℓ . Using the above notation, we would like to define a privately-correlated pseudo-random generator as a computationally private protocol¹⁰ computing the probabilistic function outputting D^ℓ (with no input), or some distribution which is indistinguishable from D^ℓ , using little communication. Specifically, we restrict the length of messages sent from the dealer to the players to be $O(\kappa)$.

Such a definition would not only be natural but also quite useful, since it gives hope to allow automatically plugging pseudo-pads in every application that uses perfect pads. However, even under our relatively benign adversary model, such a definition would be impossible to meet. To illustrate why this is the case, let n=2 and consider the simple distribution outputting a pair of identical random strings. Intuitively, any low-communication protocol for computing this distribution will reveal to P_1 a "short explanation" for the output of P_2 (which is with overwhelming probability identical to his own output), whereas if this information could be simulated based on the output of P_1 alone then one could distinguish the output of P_1 from a truly random output. In the example of additively sharing a zero, members of a collusion T will obtain a short explanation for the sum of the outputs of the remaining players (which again, cannot be simulated by viewing the T-outputs alone).

For the reason discussed above, we avoid specifying an explicit definition of the general computational primitive and replace it with an *ad hoc* definition. Namely, we define a privately-correlated pseudo-random generator with respect to a protocol or a class of protocols.

Definition 2. Let \mathcal{P} be a (perfectly or computationally) \mathcal{T} -private protocol consisting of the following two stages:

Off-line stage: The dealer sends (over secure channels) to each player P_i the *i*-th component of a random instance $y^{\ell} \leftarrow D^{\ell}$. After this stage the dealer vanishes.

On-line stage: The players interact, using their "long" random pads y^{ℓ} .

Now, let \mathcal{G} be a protocol between the dealer and the players (presumably outputting some pseudo-pads approximating D^{ℓ}), with total communication $O(\kappa)$.

We say that \mathcal{G} is a privately-correlated pseudo-random generator (PCPRG) for distribution D^{ℓ} with respect to the protocol \mathcal{P} if the protocol $\tilde{\mathcal{P}}$ obtained by replacing the off-line stage in \mathcal{P} by \mathcal{G} is computationally \mathcal{T} -private. We say that \mathcal{G} is a PCPRG for D^{ℓ} with respect to a protocol class \mathcal{C} , if it is a PCPRG with respect to every $\mathcal{P} \in \mathcal{C}$.

Using the above terminology, an ordinary pseudo-random generator may be viewed as a PCPRG in which the dealer sends a κ -bit seed to one player, who locally expands it to an ℓ -bit pseudo-random string.

¹⁰ See Appendix A for a definition of a computationally private protocol in our setting.

We now use the results of the previous section to construct, for any linear correlation D_C , a PCPRG \mathcal{G}_C with respect to a wide class of protocols meeting some mild technical requirement.

Our PCPRG \mathcal{G}_C for a linear correlation D_C is defined in a straightforward manner using the function f_M constructed in Theorem 2. Specifically, the protocol proceeds as follows. Let $G: \{0,1\}^{\kappa} \to F^{\ell}$ be any pseudo-random generator (i.e., the output of this generator is indistinguishable from a random ℓ -tuple over F). Let S denote a uniform distribution over $(\{0,1\}^{\kappa})^m$ (S represents a uniform distribution of m independent κ -bit seeds), and let $f = f_M$. In the protocol \mathcal{G}_C , the dealer first generates a seed vector $s \leftarrow S$, and sends each seed s_j to all players P_i with $j \in S_i$ (where the sets S_i are determined by the information theoretic generator f as in the previous section). Then, each player P_i evaluates $\tilde{y}_i \stackrel{\text{def}}{=} f_i(G(s))$, where $G(s) \stackrel{\text{def}}{=} (G(s_1), \ldots, G(s_m))$, using the seeds it received. The ℓ -tuple $\tilde{y}_i \in F^{\ell}$ is used as P_i 's pseudo-pad.

The next step is to define a general class of protocols \mathcal{P} for which \mathcal{G}_C is a PCPRG generating D_C . The following definition will use the notion of *simulator* as used in definitions of private protocols. We refer the reader to Appendix A for definitions of the relevant notions.

For any perfectly \mathcal{T} -private protocol \mathcal{P} , such that in the off-line stage of \mathcal{P} the dealer distributes pads generated according to D_C^{ℓ} , we define a protocol \mathcal{P}' obtained by replacing the original off-line stage by the distribution procedure induced by a corresponding \mathcal{T} -private correlation from replication generator f_M . (It can be shown that in our setting such \mathcal{P}' is also perfectly \mathcal{T} -private).

Definition 3. We say that the protocol \mathcal{P}' admits a strong simulator, if for every adversary \mathcal{A}' there exists a strong simulator \mathcal{S}' following the following two-step procedure:

- Generates, independently of its view of the inputs and the outputs, a uniformly random x_T^{ℓ} , representing pads received from the dealer in \mathcal{P}' , and outputs (the appropriate replication of) x_T^{ℓ} as the corresponding part of the simulated view;
- Proceeds arbitrarily, possibly relying on the generated view x_T^{ℓ} in addition to its view of the inputs and outputs.

Theorem 3. For any linear correlation D_C , and any protocol \mathcal{P} using D_C such that \mathcal{P}' admits a strong simulator, \mathcal{G}_C is a PCPRG with respect to \mathcal{P} .

Proof. See Appendix B.

We stress that the existence of a strong simulator in the sense of the above definition is a very mild requirement on perfectly private protocols. For instance, the simple information theoretic protocol for modular addition described in the introduction, as well as much more complicated protocols such as an implementation of [5] using linearly correlated pads, satisfy this requirement.

To see why a restriction on the protocol \mathcal{P} is at all necessary, consider a simple perfectly-private protocol \mathcal{P} in which the dealer first additively shares a long zero among the players, and then each player outputs his share. This protocols

computes, with perfect and full privacy, the *probabilistic* function outputting some distribution D_C (namely, additive sharing of zero). However, the corresponding protocol \mathcal{G}_C is not a PCPRG with respect to \mathcal{P} . (Intuitively, members of T can generate a "short explanation", which cannot be efficiently simulated, for the sum of the outputs of the other players.) Indeed, the above protocol \mathcal{P} does not admit a strong simulator, since the messages viewed by the adversary during the off-line stage depend on the adversary's view of the output.

We would like to point out a few possible extensions of the results of this section. First, pseudo-random generators can be replaced by pseudo-random functions, thereby obtaining a stronger notion of privately-correlated pseudo-random functions. Second, the dealer may be simulated by the players, resulting in private protocols under a more standard dealer-less model. In the case of a passive adversary, this can be done by picking for each seed distributed by the dealer a single player from the set of players who should receive it, and letting this player generate this seed and multicast it to the appropriate set of players. Finally, while the framework discussed in this section does not apply as is to the proactive secret-sharing application described in the introduction, our compression technique for linear correlations can be securely used in such context.

5 Concluding Remarks

While the correlation from replication paradigm seems to be the most general one could expect when pseudo-randomness is used in a black-box manner, an interesting open question is that of obtaining more efficient *direct* constructions for linear correlations and other useful types of correlations. A particularly interesting open problem is that of compressing (non-linear) correlations of the type used in [3] for allowing efficient 2-party oblivious transfer (OT) [25,11].¹¹

Acknowledgments. We wish to thank two anonymous referees for their helpful suggestions and comments. We also thank Hugo Krawczyk, Eyal Kushilevitz, and Moni Naor for related discussions.

References

- D. Beaver. Foundations of secure interactive computing. In Proceedings of CRYPTO, 1991.
- D. Beaver. Correlated pseudorandomness and the complexity of private computation. In Proc. of 28th STOC, pages 479–488, 1996.
- D. Beaver. Commodity-based cryptography. In Proc. of 29th STOC, pages 446– 455, 1997.

A seemingly related problem was studied in [2], where it is shown that an initial "seed" of κ OT's suffices to generate $O(\kappa^c)$ OT's using one-way functions alone; however, the main concern of [2] is that of weakening the required cryptographic assumptions rather than reducing the communication complexity of this task.

- D. Beaver and A. Wool. Quorum-based secure multi-party computation. In Proc. of EUROCRYPT'98, LNCS 1403, Springer Verlag, pages 375–390, 1998.
- M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computation. In *Proc. of 20th STOC*, pages 1–10, 1988.
- M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. SIAM Journal on Computing, 13(4):850–864, 1984.
- 7. B. Bollobás. Graph Theory. Springer-Verlag, 1979.
- R. Canetti. Security and composition of multi-party cryptographic protocols. Theory of Cryptography Library, Record 98-18, 1998.
- D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In Proc. of 20th STOC, pages 11–19, 1988.
- B. Chor and E. Kushilevitz. A communication-privacy tradeoff for modular addition. Information Processing Letters, 45(4):205–210, March 1993.
- S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. Comm. of ACM, 28:637–647, 1985.
- 12. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *JACM*, 33(4):792–807, October 1986.
- A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Proceedings of CRYPTO*, 1995.
- 14. M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, pages 25–34, 1997.
- Y. Ishai and E. Kushilevitz. Improved upper bound on information theoretic private information retrieval. In Proc. of 31th STOC, 1999.
- M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structures. In Proc. IEEE Global Telecommunication Conf., Globecom 87, pages 99–102, 1987.
- H. Krawczyk. Secret sharing made short. In Proceedings of CRYPTO, 1994.
- M. Luby. Pseudorandomness and Cryptographic Applications. Princeton University Press, 1996.
- F.J. Macwilliams and N.J. Sloane. The theory of error correcting codes. North Holland, 1978.
- 20. S. Micali and P. Rogaway. Secure computation. In Proceedings of CRYPTO, 1991.
- S. Micali and R. Sidney. A simple method for generating and sharing pseudorandom functions with applications to clipper-like key escrow systems. In Advances in Cryptology - CRYPTO '95, pages 185–196, 1995.
- M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In *Proc. of EUROCRYPT'99, LNCS 1592, Springer Verlag*, pages 327–346, 1999.
- R. Ostrovsky and V. Shoup. Private information storage. In Proc. of the 29th Annu. ACM Symp. on the Theory of Computing, pages 294–303, 1997.
- 24. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proc.* 10th ACM Symp. on Principles of Distributed Computation, pages 51–61, 1991.
- 25. M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
- T. Rabin. A simplified approach to threshold and proactive RSA. In Proceedings of CRYPTO, 1998.
- 27. A. Shamir. How to share a secret. Commun. ACM, 22(6):612–613, June 1979.
- 28. A. C. Yao. Theory and application of trapdoor functions. In *Proc. of the 23th Annu. IEEE Symp. on Foundations of Computer Science*, pages 80–91, 1982.

A Definitions of Private Computation

In the following we outline definitions of perfect (aka information theoretic) and computational privacy of protocols in our minimalistic model. We refer the interested reader to, e.g., [8] for more general and detailed definitions.

We consider a passive, static adversary, corrupting at the beginning of the computation some set of players $T \in \mathcal{T}$ (denoted T-players). The view of such an adversary consists of the entire joint view of the T-players. That is, this view includes the inputs and random tapes of the T-players, the messages exchanged between the dealer and T-players during the off-line distribution stage, and all messages exchanged during the on-line stage.¹²

We let a denote an input n-tuple, and g denote a (deterministic or probabilistic, single- or multi-output) function mapping the input into an output n-tuple denoted z. Notice that if g is deterministic, then a determines z. Otherwise, every input a induces some probability distribution on the output z. For every n-tuple v, v_T denotes restriction of v to its T-entries.

Let \mathcal{P} be an n-party protocol (presumably computing g). We consider two scenarios. In the real-life scenario, an adversary \mathcal{A} is exposed to the view described above induced by an execution of \mathcal{P} on some input a. In the ideal-model scenario, an output $z \leftarrow g(a)$ is picked, and a $simulator\ \mathcal{S}$, viewing only a_T, z_T (modeling what the adversary inevitably learns), has to efficiently "simulate" the real-life view of \mathcal{A} given the unknown input a and output z. We denote by $\mathbf{exec}^{\mathcal{A}}(a)$ the random variable which includes the view of the adversary \mathcal{A} on input a, concatenated with the outputs of the remaining players. ¹³ Similarly, by $\mathbf{exec}^{\mathcal{S}}(a)$ we denote the random variable including the simulated output of \mathcal{S} (on inputs a_T, z_T) along with the outputs $z_{\overline{T}}$.

We say that \mathcal{P} is a perfectly (computationally) T-private protocol computing g if for every efficient adversary \mathcal{A} (corrupting some set of players $T \in \mathcal{T}$), there exists an efficient simulator \mathcal{S} (corrupting the same set of players), such that the probability ensembles $\{\mathbf{exec}^{\mathcal{S}}(a)\}_{a\in A}$ and $\{\mathbf{exec}^{\mathcal{A}}(a)\}_{a\in A}$, are identically distributed (resp., computationally indistinguishable), where A is the (infinite) set of all input n-tuples whose n inputs are of the same length.¹⁴

B Proof Sketch of Theorem 3

Recall that the protocol $\tilde{\mathcal{P}}$ is the protocol obtained by replacing the off-line stage of \mathcal{P} or \mathcal{P}' by the protocol \mathcal{G}_C . We prove that if \mathcal{P}' admits a strong simulator, then $\tilde{\mathcal{P}}$ is computationally private.

¹² The assumption that all on-line messages are broadcasted does not compromise generality because of the existence of private channels in the off-line stage.

Setting $T = \emptyset$, this concatenation will capture the *correctness* requirement. In the case of multi-output functions, it may also capture the adversary's possibility of learning computational information about the outputs of the remaining players.

Note that this definition effectively restricts all inputs to be of the same length and uses the input length as the security parameter.

We define the following random variables for the various protocols involved. By X we denote the independent messages sent by the dealer in \mathcal{P}' (X is uniformly distributed over $(F^{\ell})^m$). By X_T^c we denote the complement of X_T , i.e. the restriction of X to the entries not viewed by players in T (notice that X_T^c is not the same as X_T), and similarly for other m-tuples. By S we denote the seeds sent in $\tilde{\mathcal{P}}$ (uniformly distributed over $(\{0,1\}^{\kappa})^m$) and by \tilde{X} the m-tuple G(S) (where G is the pseudo-random generator used by \mathcal{G}_C).

Recall that a strong simulator \mathcal{S}' for an adversary \mathcal{A}' attacking protocol \mathcal{P}' proceeds as follows. On input a_T, z_T it first outputs a uniformly random x_T (independently of its inputs), and then exactly generates the distribution of the remaining simulated view conditioned by a, z, x_T (where the probability space of the corresponding conditional view of \mathcal{A}' in the real-life protocol is over X_T^c and the coin-tosses of all players). We now construct a simulator $\tilde{\mathcal{S}}$ for an adversary $\tilde{\mathcal{A}}$ attacking the protocol $\tilde{\mathcal{P}}$. Simulator $\tilde{\mathcal{S}}$, on input a_T, z_T , proceeds as follows:

- generates random seeds s_T ;
- evaluates $\tilde{x}_T = G(s_T)$ and outputs an appropriate replication of \tilde{x}_T as the simulation of the off-line stage;
- invokes the second phase of the strong simulator S' on inputs a_T, z_T, \tilde{x}_T , and outputs the remaining simulated view.

To prove the validity of the simulator \tilde{S} constructed above, it may be conceptually convenient to define an intermediate protocol $\tilde{\mathcal{P}}'$ between $\tilde{\mathcal{P}}$ and \mathcal{P}' , obtained by replacing the pseudo-random strings \tilde{X}_T^c with perfectly random strings of the same length, X_T^c . Note that by a standard hybrid argument, X_T^c is computationally indistinguishable from \tilde{X}_T^c . We prove that a distinguisher \mathcal{D} between $\{\mathbf{exec}^{\tilde{S}}(a)\}$ and $\{\mathbf{exec}^{\tilde{A}}(a)\}$ can be turned into a distinguisher between $\{X_T^c\}$ and $\{\tilde{X}_T^c\}$. For any input a, construct a distinguisher $\hat{\mathcal{D}}_a$ as follows. On input q, sampled either according to X_T^c or \tilde{X}_T^c :

- generate random seeds s_T and evaluate $\tilde{x}_T = G(s_T)$;
- invoke the on-line stage of the protocol \mathcal{P}' on input a, using $f(\tilde{x}_T, q)$ as correlated pads, and let v_T denote the entire view of the T-players (including a_T and the seeds s_T) and z denote the protocol's outputs;
- invoke the distinguisher \mathcal{D} on $\mathbf{exec} = (v_T, z_{\overline{T}})$.

It remains to make the following key observation: If q is distributed according to X_T^c , then the random variable **exec** is distributed *identically* to the real-life view in the intermediate protocol $\tilde{\mathcal{P}}'$, which in turn is identical to $\mathbf{exec}^{\tilde{\mathcal{S}}}(a)$ (otherwise one could contradict the perfect simulation of \mathcal{S}'); on the other hand, if q is distributed according to \tilde{X}_T^c , then \mathbf{exec} is distributed identically to $\mathbf{exec}^{\tilde{\mathcal{A}}}(a)$, the real-life view in $\tilde{\mathcal{P}}$. Hence, if there exists an infinite sequence of a's on which $\mathbf{exec}^{\tilde{\mathcal{S}}}(a)$ and $\mathbf{exec}^{\tilde{\mathcal{A}}}(a)$ are distinguished by \mathcal{D} , we may use $\hat{\mathcal{D}}_a$ (with the same sequence of a's) to derive the desired contradiction to the indistinguishability of X_T^c and X_T^c .

The notation X^{ℓ} would be more appropriate than X in terms of consistency with previous notation. We omit the superscripts ℓ to avoid further notation overload.

Coding Constructions for Blacklisting Problems without Computational Assumptions

Ravi Kumar¹, Sridhar Rajagopalan¹, and Amit Sahai²

Abstract. We consider the *broadcast exclusion* problem: how to transmit a message over a broadcast channel shared by $N=2^n$ users so that *all but some* specified coalition of k excluded users can understand the contents of the message. Using error-correcting codes, and avoiding any computational assumptions in our constructions, we construct natural schemes that completely avoid any dependence on n in the transmission overhead.

Specifically, we construct: (i) (for illustrative purposes,) a randomized scheme where the server's storage is exponential (in n), but the transmission overhead is O(k), and each user's storage is O(kn); (ii) a scheme based on polynomials where the transmission overhead is O(kn) and each user's storage is O(kn); and (iii) a scheme using algebraic-geometric codes where the transmission overhead is $O(k^2)$ and each user is required to store O(kn) keys. In the process of proving these results, we show how to construct very good cover-free set systems and combinatorial designs based on algebraic-geometric codes, which may be of independent interest and application.

Our approach also naturally extends to solve the problem in the case where the broadcast channel may introduce errors or lose information.

Keywords: blacklisting, broadcast encryption, copyrights protection, error-correcting codes

1 Introduction

Consider the problem of secure communication over a broadcast channel. Such channels, for example radio, cable TV, or the Internet MBONE, are designed to transmit messages to large groups of users. Over time, however, the broadcaster may not always desire that *all* users receive its messages. Thus one can consider the problem of transmitting a message only to some small specified subset of the users, and indeed this has been considered in the literature [1]. For the case of a broadcast channel shared by numerous users, we argue that an equally, if not more, interesting problem is the complement of the above problem: how to transmit a message over a broadcast channel shared by an exponential number $N=2^n$ of users so that *all but* some specified small coalition of k excluded users can decipher the message, *even if these excluded users*

¹ IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120 {ravi,sridhar}@almaden.ibm.com

² Laboratory for Computer Science, M.I.T., Cambridge, MA 02139*** amits@theory.lcs.mit.edu

^{***} This work was done while the author was visiting IBM Almaden Research Center. Also supported in part by DOD/NDSEG fellowship and in part by DARPA grant DABT63-96-C-0018

collude with each other in an arbitrary manner. This is what we call the broadcast exclusion problem (also known as the blacklisting problem). We consider this problem as a communication problem in the most general setting possible, in particular without making any computational assumptions. We use the underlying tool of encryption only as a black box, and thus do not rely on any particular assumptions about how the encryption is implemented. Our focus is to design efficient solutions to this problem which minimize the communication overhead, while keeping the storage required by users and broadcasters feasible, i.e., polynomial in n and k. In this work, we use error-correcting codes and cover-free set systems to yield natural constructions to solve this problem, without the need for any computational assumptions. Our main result is an efficient scheme which achieves a communication blowup that depends only on the number of excluded users k. Such a result was not known previously even given computational assumptions. Our main technical tool in establishing these results is a new construction of cover-free set systems using algebraic-geometric codes.

Our Results. Arguably, the two most important parameters in a broadcast exclusion scheme are the transmission overhead (i.e., the blowup in communication) and the storage requirement for each user. In optimizing these, however, it is important to ensure that the broadcaster's storage and computation requirements also remain feasible. Our main result states that it is possible to completely avoid any dependence on n in the transmission overhead, as long as the message is of some minimal length. We provide a constructive scheme using algebraic-geometric codes where the transmission overhead is $O(k^2)$ regardless of N, and each user is required to have storage O(kn). We also present a scheme based on polynomials where the transmission overhead is O(kn) and each user has storage O(kn). Finally, we also present a scheme that depends on a randomized construction and thus requires the broadcaster to maintain a large database, but which has transmission overhead only O(k) regardless of n and in which each user has storage O(kn). All of our schemes naturally extend to provide efficient solutions to the broadcast exclusion problem where the broadcast channel may be noisy. In the process of proving these results, we show how to construct very good cover-free set systems and combinatorial designs which may be of independent interest.

We note that while it is standard when discussing communication problems to state results that are asymptotic with regard to the message length, we take care to show that our results hold even when messages are of moderate length (for example the length of what one might expect an encryption key to be in the future). And indeed, our results are interesting even when the minimum length requirement is not met.

Background. Recently, there has been growing interest and research in providing security for transmissions over broadcast mediums. The general setting for such broadcast security problems is where one has access to pairs of black-box encryption and decryption devices, which can be used to send encrypted messages over the broadcast channel and decrypt them. In the most general computational setting, for binary broadcast channels these devices could function by means of preset one-time pads, while for analog broadcast channels (such as radio or cable), these devices could encrypt based on non-computational means taking advantage of the analog nature of the communication, or other properties of the channel. In a computationally limited world, these devices

could function using private-key encryption, or using public-key encryption. In this framework, to solve broadcast security problems like the one we consider, the objective would be to distribute decryption devices to users such that the following property would hold: given some set of excluded users, one could determine a small set of encryption devices, such that none of the excluded users would have the corresponding decryption devices, while each non-excluded user would have at least one such decryption device. One could then broadcast the message using these encryption devices, and only the designated recipients would be able to decrypt it.

The two major works on blacklisting problems that have appeared in the literature have been the relatively early work of Fiat and Naor [13] and the more recent work of Wallner $\it et. al.$ [27], which have each led to several derivative works. In the framework laid out above, both these works take a similar approach: They take a universe of an exponential (in $\it n$) number of decryption devices, and distribute them according to some scheme to the users. As one might imagine, when given such a large number of decryption devices, it is quite simple to design distribution schemes which achieve the goals stated above. Indeed, [27] uses a simple balanced binary tree based scheme. By using such a large number of decryption keys, both [13] and [27] are in fact able to achieve stronger properties, for example the scheme of [27] can allow the number of excluded users $\it k$ to be arbitrary, rather than fixed in advance. To deal with this structural problem of needing so many decryption keys, for the special case of private-key based broadcast, [13], and to some extent work following [27] by Canetti $\it et. al.$ [4], make clever use of pseudo-random generators to efficiently access many pseudo-random private keys using only a few random ones.

To achieve constructions with feasible storage that do not require computational assumptions, however, these methods seem unfruitful. Thus, we take a completely different approach. Our approach yields results which compare quite favorably even against results that make computational assumptions, and also extend naturally to cover the case where the broadcast channel may introduce errors or lose information.

Our Approach. The approach we take can be interpreted from the point of view of error-correcting codes. Fundamentally, we view the broadcaster's role as creating a noisy channel—one that is very noisy indeed for the excluded users, but not so bad for the non-excluded users. Thus, by first passing the message through an appropriate error-correcting code, we are able to guarantee that non-excluded users will have enough information to reconstruct the message, while excluded users will gain no information. Interestingly, even the task of *creating* the noisy channel can be solved by using an error-correcting code with certain properties. This use of a two layer coding scheme, where the *outer code* is the one used to encode the message, and the *inner code* is used to create the noisy channel, is critical to realizing the gains in our results.

The inner code implements our noisy channel by building an interesting family of sets, which we call a (k, α) -cover-free family. This is a natural generalization of a family introduced by Erdös, Frankl, and Furedi [12]: A (k, α) -cover-free family $\mathcal{S} =$

When we say user or server "storage," we abuse terminology to mean either the physical storage of encryption or decryption devices, or in the case that these are implemented digitally, to mean the storage of the information that allows them to operate.

 $\{S_1,S_2,\ldots,S_N\}$ is one where the union of any k sets covers less than an α fraction of the elements in any other. A suitably chosen random construction can be used to design a (k,α) -cover-free family. This, however, has the disadvantage that it requires the explicit storage (space $N=2^n$) of the set system by the broadcaster. To avoid this, a careful choice of codes turns out to be rather critical. We first demonstrate our construction based on Reed-Solomon codes. We then show that algebraic-geometric codes have specific properties that make them very suitable for our purpose, yielding a construction which allows us to achieve a transmission overhead that is independent of the number of users $N=2^n$. The set systems we can construct using algebraic-geometric codes have quite strong properties, and thus may be of independent interest as well.

It is also clear from our approach that straightforward modifications to our construction (in particular, the outer code) yield blacklisting protocols that are robust against noisy broadcast channels.

A New Application: Public Broadcast Groups. We now present a new application of broadcast security protocols which falls into our general framework. Recent and proposed improvements in the quality and bandwidth of networks (see for instance IEEE 1394 [14] or the Firewire website [15]) have led to a growing interest in building and using broadcast channels to provide content, but have also sparked growing concern that digital media, such as movies, can be easily duplicated and pirated. The proposed solution to this problem is for the broadcaster to maintain a broadcast group: a group of users who share a common key which can be used to encrypt all broadcasted messages to the group. The problem arises when some set of users need to leave the group perhaps because they were caught misusing the content provided by the broadcaster. Now, the broadcaster needs to establish a new common key for all the users in the group except the set of users leaving the group. This is precisely the problem we address. Note that offending users or coalitions of users need only be blacklisted once, since all future communication (including key updates) occurs using a new common key unknown to the ousted users. Therefore it suffices to design a blacklisting protocol that can deal with a small number of excluded users at a time. This problem has motivated much study in broadcast security.

Now consider the same scenario, except where there may be many broadcasters utilizing the same broadcast channel to reach their broadcast groups. To avoid unnecessary duplication of effort, one can have a trusted facilitator set up users with public key/private key based encryption and decryption mechanisms. Then by publishing these public keys, an arbitrary number of broadcasters could use the same underlying system to maintain their broadcast groups. Users would only have to remember one set of keys aside from a single group key for each broadcast group they belong to. We call a construction which implements this a *public broadcast group scheme*. Our results apply directly to this case, and allow for efficient public broadcast group schemes where the database of public keys is of feasible size. However, since the keys being used must come from some externally specified public key encryption scheme, pseudorandom keys need not suffice. Thus, applying the results of [13] or schemes descending from the scheme of [27] without computational assumptions would require an expo-

nential number of public keys be stored for the purpose of maintaining public broadcast groups, whereas our results require only polynomially many public keys.

Prior Work. The investigation of encrypted broadcasts to selected groups of users seems to have been initiated by Berkovits [1]. A number of works have followed, focusing mainly on the problem of broadcast security for the private-key case, under the assumption that one-way functions exist. Serious study was initiated in this area by [13] by defining the term broadcast encryption, a generalization of the blacklisting problem where the issue is how to send a message excluding an arbitrary subset S of N users so that no coalition $S' \subset S$ of at most k users can decrypt the message. In this definition, however, the computation time of the broadcaster is allowed to be proportional to the size of the excluded group. The result of Fiat and Naor is based on the idea of first constructing a scheme that works for excluding a single user (using exponentially many keys) and then using multi-layered hashing techniques to break up coalitions of excluded users into single exclusions from sub-groups. Finally by taking advantage of properties of pseudo-random generators, they are able to collapse the number of keys required. The constructions from [13], when applied to solve the broadcast exclusion problem, yield (assuming the existence of one-way functions) a scheme where the transmission overhead is $O(k^2 n \log^2 k)$, and each user needs to maintain $O(kn^2 \log k)$ keys.

A second major step in this area occurred when Wallner *et. al.* [27] proposed a hierarchical scheme for the broadcast exclusion problem where k, the number of users excluded, is allowed to be arbitrary. The scheme, however, requires an exponential number $O(N=2^n)$ number of keys, but achieves a O(kn) transmission overhead and requires only O(n) keys for every user. Work of Canetti *et. al.* [4] show how to use pseudo-random generators to reduce by a constant factor the transmission overhead, but still require an exponential number of total keys. Canetti, Malkin, and Nissim [5] explore further trade-offs possible between the transmission overhead, user keys, and server keys possible by varying the scheme of [27]. Their constructions do not achieve feasible server storage, and indeed they give evidence that it is not possible to have communication overhead, user storage, and server storage all be polynomial in k and n if one requires security for arbitrary k.

A number of papers have also appeared following [13] attacking variants of the broadcast encryption and blacklisting problem under no computational assumptions. One variant that has been examined carefully is the notion of an unconditionally secure one-time broadcast encryption scheme, where the broadcasting can only be done once and all security guarantees must be unconditional. Blundo, Mattos, and Stinson [3] are able to essentially give lower bounds and upper bounds of O(1) for both the key size per user compared to the size of the message and the transmission overhead, but they require the message size be $O(N=2^n)$. Without such restrictions but still in the one-time model, Stinson and van Trung [22] give a scheme with transmission overhead approximately worse than [13] but with user key size better than the unconditional [13]. Stinson and Wei [23] improve this further, yielding a scheme which still requires an exponential number of keys, but transmission overhead roughly $O(k^2n)$. These papers on the one-time variant of blacklisting use some set systems like those we employ, but do so in conjunction with the techniques of [13] and in a way quite different from ours.

In related work, two methods—watermarking and blacklisting—are seen as the first line of defense against large-scale content piracy. The idea is for these two techniques to work in tandem. Each decoder chip will insert a watermark containing its identity into content that it decodes. Watermarking technologies have the following properties: (i) watermarks should be hard to replicate, erase, or modify, and (ii) the watermark reveals the identity of the owner of the copy. When this watermarked media is repeatedly leaked by someone, the identity of the offender is obtained from the watermark and blacklisted. For details on watermarking and similarly motivated techniques, see [8,6,11,10].

We also note that the set system we construct from algebraic-geometric codes can be used to improve some aspects of the unconditional *Tracing Traitors* construction in [6].

Organization. Section 2 discusses preliminaries including cover-free families and background on codes. Section 3 gives an overview of our approach. Section 4 gives our randomized construction. Section 5 gives the construction based on polynomials and Section 6 gives the construction based on algebraic-geometric codes.

2 Preliminaries

2.1 Cover-Free Set Systems

The notion of a *cover-free set system* is central to our approach. Let [n] denote the set $\{1, 2, \ldots, n\}$. Let N denote the number of users and B denote the set of excluded users. We first provide a formal definition of cover-free set systems

Definition 1 $((k, \alpha)$ -cover-free family). Let U be a universe, where we write n for the size of U. A family of sets $S = \{S_1, \ldots, S_N\}$ where $S_i \subseteq U$ is a (k, α) -cover-free family if for all $S, S_1, \ldots, S_k \in S$,

$$\left| S \setminus \bigcup_{i=1}^k S_i \right| \ge \alpha |S|.$$

A k-cover-free family is a (k, α) -cover-free family where α is unspecified, but strictly positive. In our applications, α will be a constant.

Constructing such a family with maximal N for a given universe size n was considered in [12]. In particular, the following theorem is established:

Theorem 1 ([12]). Given any N > 0, there are k-cover-free set systems with N sets such that $|S_i| = O(kn)$ for all i, and $n = O(k^2n)$.

However, Erdös *et. al.* establish this theorem non-constructively by showing that any maximal set system with sets of a size n/4 with the property that no two sets intersect at more than n/(4k) points will have enough sets to make N large enough for Theorem 1 to hold.

We provide an alternative constructive, although probabilistic, proof of this theorem in Section 4.

It is interesting to note that the work of [6] in the context of tracing traitors also seems to involve the need to construct cover-free set systems.

2.2 Background on Codes

Let \mathbb{F}_q be the finite field.

Definition 2 (Linear Code). An $[n, k, d]_q$ -linear code is a k-dimensional subspace of \mathbb{F}_q^n where the minimum Hamming distance between elements of this subspace is d.

The main codes we use in our constructions are Reed-Solomon and algebraic-geometric codes.

We first give the definition of Reed-Solomon codes. Let $F_{q,k}$ denote the set of polynomials on \mathbb{F}_q of degree less than k.

Definition 3 (Reed-Solomon Code). Let $x_1, \ldots, x_n \in \mathbb{F}_q$ be distinct and k > 0. The $(n, k)_q$ -Reed-Solomon code is given by the subspace $\{\langle f(x_1), \ldots, f(x_n) \rangle \mid f \in F_{q,k}\}$.

The fact that two distinct degree k polynomials can agree on at most k points can be restated as:

Property 1. The $(n,k)_q$ -Reed-Solomon code is an $[n,k,n-k+1]_q$ -linear code.

For many applications, Reed-Solomon codes are quite restrictive since they require $n \leq q$. This motivates the use of algebraic-geometric codes (or *AG codes*), which are based on algebraic curves over finite fields [17].

We now give a definition of AG codes, also known as geometric Goppa codes (see [21,20]). Let K/\mathbb{F}_q be an algebraic function field of one variable with field of constants \mathbb{F}_q and genus g. We denote the places of degree one of K by $\mathbb{P}(K)$. Suppose $|\mathbb{P}(K)| \geq n+1$. For $\alpha < n$ and Q a place of degree one, let $L(\alpha Q)$ denote the set of all $f \in K$ which have no poles except one at Q of order at most α .

Definition 4 (AG Code). Let $Q, P_1, \ldots, P_n \in \mathbb{P}(K)$ be distinct. The AG code $C(\alpha Q; P_1, \ldots, P_n)$ is given by the subspace $\{\langle f(P_1), \ldots, f(P_n) \rangle \mid f \in L(\alpha Q)\}$.

By the Riemann-Roch Theorem,

Property 2. The AG code $C(\alpha Q; P_1, \ldots, P_n)$ is an $[n, k, d]_q$ -linear code for $d \ge n - \alpha$ and $k \ge \alpha - g + 1$.

Reed-Solomon codes are straightforward to construct. There are a few AG codes in existence today (see [26,18,16]). We choose for sake of clarity (as did [20]) to focus on the family of codes given by the explicit and fairly simply Garcia-Stichtenoth function fields [16]. More details are given in Section 6.

2.3 Terminology

Although we consider our problem in a very general setting, for ease of understanding we will often talk of "keys" for encryption and decryption as if we were in the private-key setting. We use "key" to simply tie together a pair of encryption and decryption devices. Then when we talk of either encrypting or decrypting using a particular key, we mean to use the corresponding encryption or decryption device. When we talk of "giving a key to user" or a "user's key," we are referring to the decryption mechanism corresponding to that key.

2.4 Warmup: A Solution when |B| = 1

A simple solution to the broadcast exclusion problem exists when the excluded set is a singleton, i.e., $B = \{i\}$. Let n = n. We choose a set S of 2n keys and give a different subset S_i of the keys S_i , $|S_i| = n$ to each user i. Since $N < \binom{2n}{n}$, this is possible. To exclude user i, we transmit the message encrypted n times: once using each key in $S \setminus S_i$. Since each user other than i has at least one key in $S \setminus S_i$, the message can be decoded by the user. Clearly, i cannot decode the message. The storage overhead is n and the transmission overhead is n.

A central result in [13] is that for the case of private-key encryption, the transmission overhead can be reduced to O(1) for this special case. This is done via a clever use of one-way functions which allows all but the excluded user to reconstruct the session key for the transmission.

3 The Overall Construction

In this section, we describe our basic construction for the broadcast exclusion problem. Suppose we want that up to k people can be excluded at any given time. The basic idea is to have a set $K = \{k_1, k_2, \ldots, k_m\}$ of encryption/decryption keys, of which each user x gets assigned some subset S_x of u out of the m keys. When some message M is to be broadcast, it is "digested" using an error-correcting code into m smaller but redundant pieces (M_1, M_2, \ldots, M_m) , which are then encrypted according to the keys K to produce $(E_{k_1}(M_1), E_{k_2}(M_2), \ldots, E_{k_m}(M_m))$. The pieces corresponding to keys belonging to users who have been excluded are then discarded, and the remaining encrypted pieces are broadcast to all users. By decrypting the pieces corresponding to the keys that each valid user has, the user reconstructs the original message M.

For this scheme to work, we must have two properties: (i) After discarding the pieces of the message that an excluded user could intercept, we must ensure that enough pieces are left for each non-excluded user. We must try to achieve this while trying to maximize the number of possible users N we can support (i.e., the number of subsets we can make) in terms of the number of pieces m into which we split the message. This is what we call the $inner\ code$ problem. (ii) We must make sure that given the number of pieces of the message guaranteed by the inner code, each non-excluded user can reconstruct the original message. On the other hand, we do not want to waste resources, so the pieces should be as small as possible. This is what we call the $outer\ code$ problem.

The Issues. In our construction, we will try to optimize several quantities. First, we want to minimize the blowup in the communication complexity of broadcast in terms of the number of excluded users and the total number of users. We recall the best known construction has a blowup that is related to both the number of total users (which we think of as quite large—e.g., the number of cable TV subscribers) as well as the number of excluded users (which we think of as much smaller—e.g., the number of criminal TV pirating organizations). Ideally, we would like the blowup in the communication to be related only to the number of excluded users.

We also consider the problem of storage. For the user, we want to minimize the number of keys that each user must store. We would also like to avoid having the broadcaster store a large database with information for each user. Rather, we would like the broadcaster to be able to efficiently generate the relevant information about a user given only his identity. Although we do give one randomized scheme that requires a large database, we focus mainly on the problem where storage requirements of the broadcaster are not to depend polynomially on the number of users.

The Methods. We will focus on the inner code and the outer code separately. First, we observe that the outer coding problem can be solved trivially by a simple repetition code—i.e., the message M is simply repeated so $M = M_1 = M_2 = \cdots = M_m$.

The Inner Code. Now, if we can simply solve the inner coding problem so that at least one of the message pieces can be decrypted by each good user after all the message pieces corresponding to the k excluded users have been removed, then our construction will work. Constructing such set systems is precisely the problem Erdös *et. al.* considered [12], i.e., finding families of finite sets such that no set is covered by the union of k others. In Section 4, we describe a randomized construction that matches the lower bound of [12]. In Sections 5 and 6, we give constructions based on error correcting codes. When building an inner code, not just any "good" error correcting code will do. The fact that we want sets to have very small intersections corresponds to the underlying code having good minimum distance at very low rates. Polynomial codes have this property. To optimize transmission overhead in conjunction with the outer code, however, it will turn out that we want the sets in our set system to be fairly large. This corresponds to underlying field of the code being small. Polynomial codes fail this property but this is precisely what algebraic-geometric codes were designed for. These intuitions are made formal in Sections 5 and 6.

The Outer Code. The purpose of the outer code is to eliminate waste in the broadcast transmission, i.e., to allow users to fully utilize the (limited) information they are able to decrypt during a broadcast. For this reason, the outer code should be a good low-rate erasure code. In this exposition, we will focus on polynomial-based codes, which have the advantage of being *perfect* erasure codes in that no information is wasted. We could also employ other erasure codes with smaller alphabet sizes to minimize how long the message being transmitted has to be, but we omit the details of these issues in this extended abstract.

We define an $[n,k,m]_q$ (constructive) erasure code to be a (polynomial-time) function $C:\mathbb{F}_q^k\mapsto\mathbb{F}_q^n$ such that there exists a (polynomial-time) function $D:\bar{\mathbb{F}}_q^n\mapsto\mathbb{F}_q^k$, where $\bar{\mathbb{F}}=\mathbb{F}\cup\{\bot\}$, such that: For all $v\in\mathbb{F}_q^k$, if $u\in\bar{\mathbb{F}}_q^n$ is such that u agrees with C(v) on at least m places, and is \bot elsewhere, then D(u)=v. In other words, one can erase up to n-m positions of the codeword and still reconstruct the original message. Clearly, the best one can hope for is to have m=k. Such an erasure code is called a perfect code.

Polynomials. Here we recall that polynomials allow one to construct perfect $[n, k, k]_q$ erasure codes for any $k \leq n \leq q$. This is called the *Reed-Solomon code*. Given a vector $v = (v_0, v_1, \ldots, v_{k-1}) \in \mathbb{F}_q^k$, we construct the polynomial $p_v(x) = v_0 + v_1 x + \ldots + v_{k-1} x^{k-1}$. Let e_1, e_2, \ldots, e_n be some canonically chosen elements of \mathbb{F}_q . Then $C(v) = v_0 + v_1 x + \ldots + v_{k-1} x^{k-1}$.

 $(p_v(e_1), p_v(e_2), \ldots, p_v(e_n))$. In the decoding scenario, we are given k pairs of the form $(e_i, p_v(e_i))$. Since we know $\deg(p_v) < k$, simple polynomial interpolation over \mathbb{F}_q will efficiently allow us to reconstruct the coefficients of p_v , yielding the original vector v.

Communication. It follows from the definition that if C is an $[n, k, m]_q$ erasure code, then the length of C(v) is $n \log q$ bits, whereas the length of the original message v is $k \log q$ bits. Hence the communication blowup is n/k.

Fault Tolerance. We see immediately that by changing the parameters of the outer code, we can easily make all our construction tolerant to faults in the broadcast channel. If the channel can drop packets, then simply by using the erasure correction properties we have already discussed, robustness can be achieved. If the channel also corrupts data, one can use a code with both erasure and error-correcting capabilities. This is possible for example with Reed-Solomon codes [2]. We omit the details of how these parameters should be chosen to achieve different levels of robustness in this extended abstract.

4 A Randomized Construction

We now describe a randomized construction of a broadcast exclusion scheme that works with high probability.

The Inner Code. We begin with a randomized construction that with high probability yields a set system with the desired properties. This construction matches the lower bound given in [12]. Let the universe U=[m]. We wish to construct a (k,α) -cover-free family $\mathcal S$. For sake of clarity, we give our construction only for $\alpha=1/2$ (which we will use later), although the construction can easily be seen to work for any constant α . Also, for sake of clarity, we give present an analysis that is not tight in the calculation of some of the constants, but captures the essence of what we can achieve in this randomized construction given below:

Partition U into m/2k sets $U_1, U_2, \ldots, U_{m/2k}$ each of size 2k. Pick each of N sets as follows: Pick $x_i \in U_i$ uniformly for each i, and let $S = \{x_1, \ldots, x_{m/2k}\}$.

We now show the following theorem:

Theorem 2. There exists a universal constant c such that for any t > 0, if

$$N = \exp\left(\frac{cm}{k(k+1)} - \frac{t}{k+1}\right),\,$$

then the probability that the above set system is not (k, 1/2)-cover free is at most $\exp(-t)$.

Proof. Without loss of generality, let $k \geq 2$. Fix sets $S, S_1, S_2, \ldots, S_k \in S$. For any $x \in U_j$ for some j, we have

$$\Pr\left[x \notin \bigcup S_i\right] = \left(1 - \frac{1}{2k}\right)^k \ge \frac{9}{16}.$$

Hence,

$$\operatorname{E}\left[\left|S \cap \left(\bigcup S_i\right)\right|\right] = \left(1 - \left(1 - \frac{1}{2k}\right)^k\right) \cdot |S| \le \frac{7}{16} \cdot |S|.$$

By Chernoff bounds [7],

$$\Pr\left[\left|S \cap \left(\bigcup S_i\right)\right| > \frac{|S|}{2}\right] \le \exp\left(2 \cdot \left(\frac{1}{16}\right)^2 \cdot |S|\right) = \exp\left(\frac{m}{256k}\right).$$

By the union bound, the probability that this occurs for any choice of S, S_1, S_2, \ldots, S_k - i.e., that S does not satisfy the condition given in the theorem—is at most

$$N^{k+1} \cdot \exp\left(\frac{m}{256k}\right)$$
.

Letting N as in the statement of the theorem, we see that this probability is at most $\exp(-t)$.

It is immediate that if we are willing to tolerate the broadcaster having a large database to store the keys assigned to each user, this randomized construction, when used in conjunction with a trivial repetition outer code, yields a broadcast exclusion scheme with a blowup in communication of $O(k^2n)$.

The Outer Code. We will now see how to use an outer code to reduce this communication blowup to O(k). The inner code construction above shows that to support N users, we need to have $m=O(k^2n)$ total keys and total messages sent, with each user receiving m/2k=O(kn) keys. Moreover, the construction guarantees that with high probability, even with k users blacklisted, every good user will be able to decrypt at least m/4k of the messages.

Hence, we may use a $[m,m/4k,m/4k]_q$ constructive erasure code as the outer code, such as the polynomial construction given earlier. Here, we pick the field size $q \geq m$. Thus, if we assume the message is at least $(m/4k) \cdot \log q = \Omega(kn \log k \log n)$ bits long (which is reasonable even if the message is a single strong cryptographic key), the communication blowup is reduced to simply 4k. Note that this bound on the message size can be even improved further using more sophisticated erasure codes. The details are omitted from this extended abstract.

5 Construction Based on Polynomials

We now give a deterministic construction based on polynomials.

The Inner Code. Let m be such that $q=\sqrt{m}$ is a prime power and let $\mathbb{F}_q=\{u_1,\ldots,u_q\}$ be an enumeration of the field. Let $F_{q,d}$ be the set of polynomials on \mathbb{F}_q on degree at most d. Let the universe $U=\mathbb{F}_q^2, |U|=m$ and let d=q/2k. Let the set system $\mathcal{S}=\{S_f\mid f\in F_{q,d+1}\}$, where $S_f=\{\langle u_1,f(u_1)\rangle,\ldots,\langle u_q,f(u_q)\rangle\}\subset U$. We denote the size of \mathcal{S} by N.

We now show the following theorem:

Theorem 3. The set system S is (k, 1/2)-cover free with

$$N = \exp\left(\frac{\sqrt{m}\log m}{4k}\right).$$

Proof. Notice that the construction of S corresponds exactly to an $(q, d+1)_q$ -Reed-Solomon code. Since two distinct degree d polynomials can agree on at most d points, we see that $|S_i \cap S_j| \leq d$ for any distinct $S_i, S_j \in S$. So, for any $S_i, S_i, \ldots, S_k \in S$,

$$\left| S \setminus \bigcup_{i=1}^{k} S_i \right| \ge q - k \left(\frac{q}{2k} \right) = \frac{q}{2},$$

as desired. Since $|F_{q,d}| = q^{d+1}$, the bound on N follows.

This scheme, when combined with a trivial repetition outer code, yields a deterministic broadcast exclusion scheme with a blowup in communication of $O(k^2 \log^2 N)$.

As Theorem 3 shows, the value of N is sub-optimal. In Section 6, we show how to improve this construction using the more powerful AG codes.

The Outer Code. As in Section 4, we can use an outer code to significantly reduce the communication blowup. The inner code construction above shows that to support N users, we need to have $m = O(k^2 \log^2 N)$ total keys and total messages sent, with each user receiving q = O(kn) keys. Moreover, the construction guarantees that even with k users blacklisted, every good user will be able to decrypt at least q/2 of the messages.

Let $q' \ge q^2$. We can use an $[m, q/2, q/2]_{q'}$ constructive erasure code as the outer code, such as the polynomial construction given earlier. Thus, if we assume the message is at least $q \log q = \Omega(k \log kn \log n)$ bits long, the communication blowup is just 2m/q = 2q which is O(kn).

6 Construction Based on AG Codes

As we can see, the fundamental limitation of the polynomial-based scheme given in Section 5 was that the size of the sets had to be much smaller than the total number of possible elements. This was because the number of points where we could evaluate the polynomials was bounded by the field size. It is precisely this problem that algebraic-geometric (Goppa) Codes were designed to address. We will show that this property allows one to construct very good set systems using them as well. There are a few Goppa codes in existence today that would address our problems [26,18,16], we chose for sake of clarity (following [20]) to focus on the family of codes given by the explicit and fairly simply Garcia-Stichtenoth function fields [16].

Garcia and Stichtenoth [16] give an explicit construction of function fields F_n extending \mathbb{F}_{q^2} . The main theorem of the paper is that F_n has at least $s(n)+1\stackrel{\mathrm{def}}{=} q^{n-1}(q^2-1)+1$ places of degree one and genus at most $g(n)\stackrel{\mathrm{def}}{=} q^{n-1}(q+1)$. Thus, the asymptotic ratio of the number of places of degree one to the genus is q-1, attaining the Drinfeld-Vladut upper bound. Indeed, such function fields were known to

exist by the celebrated Tsfasman-Vladut-Zink theorem [26], and Manin and Vladut [18] showed that the corresponding Goppa codes are constructible in polynomial time. The function fields underlying the work of [26], however, are far from being explicit. The main contribution of [16]— can be seen as giving a simple construction of function fields attaining the Drinfeld-Vladut bound. Indeed, we can describe the construction here: Let $F_1 = \mathbb{F}_{q^2}(x_1)$, the rational function field over one variable x_1 . For $n \geq 2$, let $F_n = F_{n-1}(z_n)$, where z_n satisfies the equation $z_n^q + z_n = x_{n-1}^{q+1}$ and for $n \geq 3$, $x_{n-1} \stackrel{\text{def}}{=} z_{n-1}/x_{n-2}$.

By Property 2, for every $n \geq 3$ and $\alpha \leq s(n)$ the Garcia-Stichtenoth construction yields an $[s(n),k,d]_{q^2}$ -linear code $C=C(\alpha Q;P_1,\ldots,P_s(n))$, where $d\geq s(n)-\alpha$, $k\geq \alpha-g(n)+1$ and $Q,P_1,\ldots,P_s(n)\in \mathbb{P}(F_n)$, i.e., are distinct places of degree one in F_n .

The Inner Code. Let q be chosen so that $k = \lfloor q/6 \rfloor$, and let $n \geq 3$. Let $Q, P_1, \ldots, P_{s(n)} \in \mathbb{P}(F_n)$ be distinct places of degree one. Let the universe $U = \{P_1, \ldots, P_{s(n)}\} \times \mathbb{F}_{q^2}$, and we denote the size of U by $m = s(n)q^2$. Finally, let $\alpha = g(n) + m/q^3$. Then we define the set system $\mathcal{S} = \{S_f \mid f \in L(\alpha Q)\}$, where $S_f = \{\langle P_1, f(P_1) \rangle, \ldots, \langle P_{s(n)}, f(P_{s(n)}) \rangle\} \subset U$. We denote the size of \mathcal{S} by N.

Theorem 4. The set system S is (k, 1/2)-cover free with

$$N = \exp\left(O\left(\frac{m\log k}{k^3}\right)\right).$$

Proof. The construction of S above obviously corresponds to the AG code $C = C(\alpha Q; P_1, \ldots, P_s(n))$. Thus, because C is a code with minimum distance at least $s(n) - m/q^3 - g(n)$, it follows that $|S_i \cap S_j| \leq m/q^3 + g(n)$ for any distinct $S_i, S_j \in S$. So, for any $S_i, \ldots, S_k \in S_i$,

$$\left| S \setminus \bigcup_{i=1}^{k} S_{i} \right| \ge s(n) - k \left(\frac{m}{q^{3}} + g(n) \right)$$

$$= q^{n+1} - q^{n-1} - \frac{2q^{n+1} - q^{n} - q^{n-1}}{6}$$

$$\ge \frac{q^{n+1} - q^{n-1}}{2} + \frac{q^{n+1} - q^{n}}{6}$$

$$\ge \frac{s(n)}{2}$$

as desired. Since $N=|L(\alpha Q)|=(q^2)^{\alpha-g(n)}=(q^2)^{m/q^3}$, the bound on N follows.

This scheme, combined with the trivial repetition outer code, yields a deterministic broadcast exclusion scheme with a blowup in communication of $O(k^3n)$.

The Outer Code. We will now show how to use an appropriate outer code to reduce the communication blowup to just $O(k^2)$, eliminating dependence on N and almost

matching the randomized construction of Section 4. The inner code construction above shows that to support N users, we need to have $m = O(k^3n)$ total keys and total messages sent, with each user receiving $m/q^2 = O(kn)$ keys. Moreover, the construction guarantees that even with k users blacklisted, every good user will be able to decrypt at least $m/2q^2$ of the messages.

Let $q' \geq m$ be a prime power. We can use an $[m, m/2q^2, m/2q^2]_{q'}$ constructive erasure code as the outer code, such as the polynomial construction given earlier. Thus, if we assume the message is at least $(m/q^2) \cdot \log q' = \Omega(k \log k n \log n)$ bits long, the communication blowup is just $2q^2$ which is $O(k^2)$, independent of the number of users N.

References

- S. Berkovits. How to broadcast a secret. Proc. of EUROCRYPT, Springer LNCS 547:535–541, 1991.
- E. R. Berlekamp. Bounded distance + 1 soft decision Reed-Solomon coding. *IEEE Trans. on Information Theory*, 42:704–720, 1996.
- C. Blundo, L. F. Mattos, and D. R. Stinson. Generalized Beimel-Chor schemes for broadcast encryption and interactive key distribution. *Theoretical Computer Science*, 200(1-2):313–334, 1998.
- R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. *Proc. IEEE INFOCOM'99*.
- R. Canetti, T. Malkin, and K. Nissim. Efficient communication-storage tradeoffs for multicast encryption. *Proc. EUROCRYPT 99*, to appear.
- 6. B. Chor, A. Fiat, and M. Naor. Tracing traitors. Proc. CRYPTO, pp. 257-270, 1994.
- H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.
- I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon. A secure, robust watermark for multimedia. Workshop on Information Hiding. Newton Institute, University of Cambridge, 1996.
- 9. I. J. Cox and M. L. Miller. A review of watermarking and the importance of perceptual modeling. *Proc. Electronic Imaging*, 1997.
- C. Dwork. Copyright? Protection? The Mathematics of Coding, Extraction, and Distribution: IMA Volumes in Mathematics and its Applications, 107, 1996.
- 11. C. Dwork, J. Lotspiech, and M. Naor. DigitalSignets: Self-enforcing protection of digital information. *Proc.* 28th ACM Symposium on Theory of Computing, pp. 489–498, 1996.
- 12. P. Erdös, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of *r* others. *Israel Journal of Mathematics*, 51:79–89, 1985.
- 13. A. Fiat and M. Naor. Broadcast encryption. *Proc. CRYPTO*, pp. 480–491, 1993.
- 14. IEEE. 1394 Specifications. Available from customer.service@ieee.org.
- 15. http://firewire.org/.
- A. Garcia and H. Stichtenoth. A tower of Artin-Schreier extensions of function fields attaining the Drinfeld-Vladut bound. *Invent. Math.*, 121:211–222, 1995.
- 17. V. D. Goppa. Codes on algebraic curves. Soviet Math. Dokl., 24:170-172, 1981.
- 18. Y. I. Manin and S. G. Vladut. Linear codes and modular curves. *J. Soviet Math.*, 30:2611–2643, 1985.
- S. Mittra. Iolus: A framework for scalable secure multicasting. Proc. ACM SIGCOMM Conference: Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 277–288, 1997.

- M. A. Shokrollahi and H. Wasserman. Decoding algebraic-geometric codes beyond the error-correction bound. *Proc. 30th ACM Symposium on Theory of Computing*, pp. 241– 248, 1998.
- H. Stichtenoth. Algebraic Function Fields and Codes. Universitext, Springer-Verlag, 1993.
- 22. D. R. Stinson and T. van Trung. Some new results on key distribution patters and broadcast encryption. *Designs, Codes, and Cryptography*, to appear.
- D. R. Stinson and R. Wei. An application of ramp schemes to broadcast encryption. Manuscript.
- D. R. Stinson and R. Wei. Key preassigned traceability schemes for broadcast encryption. Manuscript.
- 25. L. Trevisan. Constructions of near-optimal extractors using pseudorandom generators. 31st ACM Symposium on Theory of Computing, pp. 141–148, 1999.
- 26. M. A. Tsfasman, S. G. Vladut, and Th. Zink. Modular curves, Shimura curves, and Goppa codes better than the Varshamov-Gilbert bound. *Math. Nachrichten*, 109:21–28, 1982.
- 27. D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. ftp://ietf.org/internet-drafts/draft-wallner-key-arch-01.txt

An Information Theoretic Analysis of Rooted-Tree Based Secure Multicast Key Distribution Schemes

R. Poovendran* and J. S. Baras **

Dept. of Electrical Engineering & Institute for Systems Research University of Maryland, College Park, MD 20742, USA

Abstract. Several variations of rooted tree based solutions have been recently proposed for member revocation in multicast communications [18, 19, 20, 21]. In this paper, we show that by assigning probabilities for member revocations, the optimality, correctness, and the system requirements of some of these schemes [18, 19, 20, 21] can be systematically studied using information theoretic concepts. Specifically, we show that the optimal average number of keys per member in a rooted tree is related to the entropy of the member revocation event. Using our derivations we show that (a) the key assignments in [18, 21, 20, 19] correspond to the maximum entropy solution, (b) and direct application of source coding will lead to member collusion (we present recently proposed solutions [21, 20] as examples of this) and a general criteria that admits member collusion. We also show the relationship between entropy of member revocation event and key length.

1 Introduction

Recent research in multicast communication has created the possibility of several new business applications with potential need for secrecy and integrity of the communication ([18]-[26]). Potential commercial applications are stock quotes, special sporting events, Internet news and multimedia related applications such as conferences, etc. Due to the distributed nature and the involvement of more than two parties in these applications, there are some unique security related issues that are relevant only to secure multicast communications. Issues that pose significant challenges are: (a) preserving the integrity and secrecy of the communication, (b) dealing with the dynamic nature of the group membership, (c) being able to secure the intermediate nodes such as the routers, (d) graceful failure of administrative nodes, and (e) member addition/deletion.

In secure multicast communication, all the members share a common Session Encrypting Key (SK). Members of the group should also have Key Encrypting Key(s) (KEK) that can be used to update the SK in the event of membership

^{*} Email: radha@isr.umd.edu, http://www.ece.umd.edu/~radha

^{**} Email: baras@isr.umd.edu, http://www.ece.umd.edu/~baras

change due to any of the following reasons (a) a new member admission, (b) expiration of the SK, (c) member compromise, (d) voluntary leave, and (e) member revocation. When the membership increases, the SK may be changed to protect the back traffic and in all other cases, the SK is changed in an effort to protect future traffic. Developing efficient key update schemes while attempting to prevent member collusion and allowing the group center to perform member revocation has been the focus of several recent efforts [13, 23, 27, 28, 18, 19, 11, 21, 20]. We review two extremes of the non-tree based methods below and then focus on the rooted tree based scheme in the rest of the paper. A list of relevant papers are given in the reference section. The group size is denoted by N throughout this paper.

1.1 Non-tree Based Key Distribution Approaches

The secure group communication requires KEKs to securely distribute the updated SK. If every member has an individual public key, for a group consisting N members, the SK update will involve $\mathcal{O}(N)$ encryption by the GC. The linear increase of the required number of encryptions in group size is not suitable for very large scale applications common in Internet, due to the amount of computational burden on the GC.

A simple way to reduce the number of encryption by the GC at the time of SK update is to provide a common KEK to all the members of the group as suggested in [23]. If the SK is to be updated due to its lifetime expiration the GC can perform a single encryption and update all the group members. If the SK is to be updated due to a new member admission, before admitting the new member, the GC may choose a new SK and the future KEK, encrypt both using the current KEK and update all the members. The newly admitted member is given the new SK and the KEK separately. However, this approach fails to support the secure communication if a single member is to be deleted/revoked. Since the whole group, including the deleted/revoked member share a single KEK, a revoked member will have access to all future key updates. Hence, this approach doesn't provide an efficient recovery mechanism for the valid members in the presence of single member failure.

In [13, 18], an approach that partition the set of keys assigned to a member into two groups was proposed. One of these sets is called the complement set and contains keys that are not distributed to a particular member. If each member has a unique complementary set, this set can be used for key updates in the event the corresponding member is revoked. The GC associates a KEK and a member in a one-to-one manner. If there are N members in the group, there will be N KEKs each representing a single member. The GC then distributes these N KEKs such that a member is given all the KEKs except the one associated with him/her. Hence, the complementary set contains a single KEK for each member. If the GC wants to delete/revoke a member, it needs to broadcast only the member index to the rest of the group. Since all members except the revoked one has the associated KEK of the revoked member, they can use that KEK for the future SK updates. This approach requires only one encryption at

the GC and allows the GC to update the SK under single member compromise. In fact this approach seem to allow even multiple member deletion/revocation. However, considering the complementary sets of any two members reveals that all the KEKs of the group are covered by the KEKs held by any two members. Hence, any two deleted/revoked members can collaborate and have access to all future conversations. Thus, under user collusion, this key scheme does not scale beyond two members. Thus the scheme doesn't have perfect forward secrecy under collusion of revoked members. This approach requires KEK storage that scales as $\mathcal{O}(N)$.

Above mentioned schemes are two extremes of KEK distribution. Depending on the degree of user collusion, a large variety of key management schemes with different number of KEKs per member can be generated.

Recently a series of papers utilizing rooted-trees for key distribution have been proposed to minimize the storage at the group controller and the members while providing a reduction in the amount of encryptions required to update the session key [11, 12, 18, 19, 20, 21, 28]. Some efficient schemes based on oneway functions also have been used on the trees for member revocation. Many of these tree based schemes seem to present different solutions to the problem with different values for the required keys to be stored at the GC and the user node. Aim of this paper to unify these results and analyze them.

1.2Organization of the Paper

In section 2, we review the seminal work on currently known rooted tree based revocation schemes [18, 19]. We show that the approach in [18, 19, 20, 21] is related to well-known prefix coding in the rooted trees. In section 3, we define an appropriate notion of member revocation event and the associated probabilities. Using this probabilistic modeling, we show that the optimal average number of keys per member (and hence the average number of keys to be updated at the time of a member revocation) is equal to the entropy of the member revocation event. We further show that the optimal strategy is to assign a member with higher revocation probability less number of keys. Section 4 shows that although the basic idea of information theory can be used to find the optimal number of keys to be given to a member, trying to directly use optimal coding scheme, namely the Huffman coding, for key allocation will lead to member collusion. In order to justify our claims, we use the results in [20, 21] to show that the approaches in these two schemes can be interpreted as the Huffman coding and then show that both the schemes can be broken if two appropriate members collude or are compromised, regardless of the group size N. Using the source coding part of the information theoretic approach in analyzing the key allocations in [20, 21] allows us to characterize the collusion problem with these approaches for any N and D. In section 5 we show how to use the entropy of member revocation event to average hardware requirements of the key generating system and bound the length of the key that can be generated.

2 Review of the Logical Key Tree

Given a set group of N members and a number base D, $\log_D N$ D-ary digits are sufficient to uniquely index each of the N members in base D. This D-ary representation can also be viewed as a rooted tree representation with each member being the leaf of a D-ary tree of depth $\log_D N$. (For illustrations, we use D=2 in the figures, leading to binary trees).

2.1 Distribution of Keys on the Tree

As a concrete illustration, figure 1 presents a KEK distribution based on a binary rooted tree for 8 members. In this approach, each leaf of the tree represents a unique member of the group; i.e. the leafs are in one-to-one correspondence with members. Each node of the tree represents a key. The set of keys along the path from the root to a particular leaf node are assigned to the member represented by that leaf node. For example, member M_1 in figure 1 is assigned KEKs $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$.

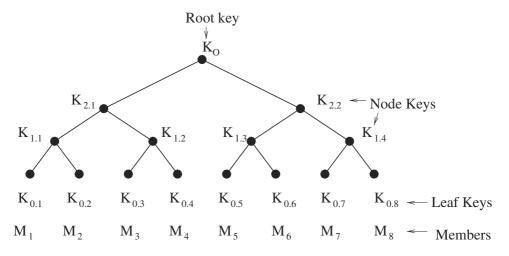


Fig. 1. The Logical Key Tree of [11, 18, 19, 20, 21]

If there is no member deletion/revocation or compromise, the common KEK denoted by K_O can be used to update the SK for all the members. The tree based structure also induces a natural hierarchical grouping among the members. By logically placing the members appropriately, the GC can choose the appropriate keys and hence selectively update, if need be, the keys of the group. For example, in figure 1, members M_5 , M_6 , M_7 , and M_8 exclusively share the key $K_{2.2}$. The GC can use the key $K_{2.2}$ to selectively communicate with members M_5 , M_6 , M_7 , and M_8 . Hence, the local grouping of the members and the keys shared on the tree may be decided by the GC based on application specific needs. In order to

be able to selectively disseminate information to a subset of group members, the GC has to ensure that the common key assigned to a subset is not assigned to any member not belonging to that subset. Using the notation $\{m\}_K$ to denote the encryption of m with key K, and the notation $A \longrightarrow B : \{m\}_K$ to denote the secure exchange of message m from A to B, the GC can selectively send a message m to members five through eight by the following transmission: $GC \longrightarrow M_5, M_6, M_7, M_8 : \{m\}_{K_{2,4}}$

If the key $K_{2.2}$ is invalidated due any reason, the GC needs to update the key $K_{2.2}$ before being able to use a common key for members M_5, M_6, M_7 , and M_8 . It can do so by first generating a new version of $K_{2.2}$, and then performing two encryptions, one with $K_{1.3}$ and the other with $K_{1.4}$. The following two messages are needed to update key $K_{2.2}$ to the relevant members of the group.

 $GC \longrightarrow M_5, M_6 : \{K_{2.2}\}_{K_{1.3}}$ $GC \longrightarrow M_7, M_8 : \{K_{2.2}\}_{K_{1.4}}$

2.2 Member Revocation in Rooted Trees

From now on, we will use the term keys to denote SK or KEKs unless there is a need for clarification. Since the SK and the root KEK are common to all the members in the multicast group, they have to be invalidated at each time a member is revoked. Apart from these two keys, all the intermediate KEKs of the revoked member need to be invalidated. In the event there is bulk member revocation, the GC has to

- Identify all the invalid keys,
- Find the minimal number of valid keys that need to be used to transmit the updated keys.

For an arbitrary tree that may not hold members in all the leafs these two problems need to be solved by exhaustive search. The principle behind the member revocation is discussed below by an example.

Member M_1 in figure 1 is indexed by the set of four keys $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$. Revoking M_1 is equivalent to invalidating these four keys, generating four new keys, and updating these keys of the appropriate valid members. When M_1 is revoked, the following key updates need to be performed: (a) all member need new K_O , (b) members $M_2 - M_4$ need to update $\{K_{2.1}\}$, (c) members $M_3 - M_4$ need to update $\{K_{1.2}\}$, and (d) member M_2 needs to update $\{K_{1.1}\}$.

Following observations can be made towards the rooted tree based key distributions.

- Since each member is assigned $(2 + \log_d N) = \log_d N d^2$ keys, deletion of a single member requires $(2 + \log_d N)$ keys to be invalidated.
- Since there are $(1 + \log_d N)$ nodes between the root and a leaf and $\log_d N$ nodes are shared with other members, and for each common node one encryption is required, the GC needs to perform a total of $\log_d N$ encryptions.
- For a d-ary tree with depth $h=\log_d N$, the GC has to store $1+1+d+d^2+\cdots+d^h=\frac{d(N+1)-2}{(d-1)}$ number of keys. Setting d=2 leads to the binary tree

for which the required amount of storage works out to be $\frac{2(N+1)-2}{2-1}=2N$. This result can be independently checked by noting that a binary tree with N leafs has 2N-1 nodes. Hence the GC has to store the SK and (2N-1) KEKs, leading to 2N keys that need to be stored.

In [20, 21], binary rooted tree based key distributions which require the GC to store a total of $2\log_2 N$ distinct keys were proposed. The generalized version of this result requires $d\log_d N$ keys to be stored at the GC. Each member needs to store only $(2 + \log_d N)$ keys in this scheme. However, the number of keys to be updated remain at $\log_d N$ as in [18, 19]. Hence, at first glance, the results in [21] seem to reduce the storage requirements at the GC by

$$\frac{d(N+1)-2}{d-1} - d\log_d N = \frac{d(N+1-(d-1)\log_d N)-2}{(d-1)}$$
 (1)

number of keys without increasing the key storage requirements at the end user node.

In the next section we present our analytical formulation to study these models in a systematic manner.

2.3 Preliminary Observations: Properties of Keys on Rooted Trees

We use the approach that uses Kraft inequality in this section since we intend to derive the optimal number of keys for individual members as well. We note that the approach based on [4] can be used if we are interested in average performance analysis for a given rooted tree based scheme.

Relationship between Prefix Coding and Member Revocation Unless the set of keys held by each member differ by at least one key, the group center will not be able to successfully update the keys of the group after revoking a member. More importantly, the keys held by any member should not form a subset of the keys held by another member in rooted-tree. This is equivalent to the condition that no internal node should index a member and no two members should be indexed by the same leaf. If we view the concatenation of keys given to a member as a Key Index (KID), then each of the member should have a distinct KID. In our definition, we consider any permutation of the KID elements as identical to the original KID. Hence there are L! equivalent KIDs for a member with L KEKs. This distinction is important in dealing with the user collusion. (The KID for member 1 in figure 1 is represented by $K_{2,1}K_{1,1}K_{0,1}$.)

To illustrate the collusion problem by an example, let members i, j, and k have sets of keys denoted by S_i, S_j and S_k respectively, where, $S_i = \{K_0, K_1, K_2\}$, $S_j = \{K_0, K_1, K_2, K_3, K_4, K_5\}$, and $S_k = \{K_3, K_4, K_5\}$, respectively. Clearly, $S_i \subset S_j$, and $S_k \subset S_j$. In the event member j is compromised, all the keys of members i, and k are to be treated compromised. If the group center tries to use any one of the keys in the set S_i or S_k to encrypt the new set of keys for member i, and k, revoked member j can decrypt and access all the communication since

it has all the keys of i and k. In such cases, removal of one or more members, who have all the keys of one or more valid members, will compromise the integrity of the key updates and the privacy of the entire future communication.

Choosing unique KIDs on the rooted-trees is equivalent to the requirement that is equivalent to the statement in source coding that no codeword should be a prefix to another codeword. This condition can be restated as the KID corresponding (or a set of keys assigned) to a member should not be a prefix to the KIDs corresponding (or a set of keys assigned) to any other member. If not, mapping between the set of keys and the members will not be unique. Assigning a unique prefix code to each member on the tree leads to the following (more or less) well known important theorem that will be used later in this paper.

Theorem 1. Kraft Inequality for KIDs

For a D-ary Logical key tree with N members and a codeword generated by the concatenation of a set of keys such that no two members have the same codewords (set of keys) and the codeword of anyone member is not a prefix of the codeword of any other member, if we denote the codeword length (number of keys held by that member) for member i by l_i , the sequence $\{l_1, l_2, \dots l_N\}$ satisfies the Kraft inequality given by

$$\sum_{i=1}^{i=N} D^{-l_i} \le 1. (2)$$

Conversely, given a set of numbers $\{l_1, l_2, \dots l_N\}$ satisfying this inequality, there is a rooted tree that can be constructed such that each member has a unique set of concatenated keys with no-prefixing.

Proof: Standard and can be found in [2, 3].

Relationship Between Member Collusion and Codewords In a group of more than two members, member collusion needs to be prevented to preserve integrity of the group communication. We illustrate collusion using the same set of members above. If the group center needs to revoke members i and k, and update the keys for member j, revoked members i and k can collude and construct the set S_j since $S_j = S_i \cup S_k$. Hence, any rooted-tree structure should not permit a member to have a set of keys is a subset of the keys of other members or can be obtained as a concatenation of keys of other members. In this example, we could concatenate the key sets of members i and k to get the key set of member j. We note that a variation of rooted-tree presented in [20, 21] does suffer from member collusion. In a later section we will prove this claim and characterize the type of collusion of rooted trees [20, 21] using information theory. We note however that the codeword representation of the keys is not enough to characterize all types of collusions. We will discuss this point in the section 4.

3 Probabilistic Modeling

Since the key updates are performed in response to revocation of members, statistics of member revocation events, are very appropriate for system design and performance characterization. We denote p_i as the probability of revocation of member i with $\sum_{i=1}^{i=N} p_i = 1$. If the revocation were to have zero probabilities, then there is no issue of revocation of keys in the first place at all³. Hence, this assignment of probabilities is consistent with the motivation of key revocation.

3.1 Optimizing the Rooted Tree Structure: Optimal Codeword Length Selection

Optimization of average number of keys per members with the length of the KIDs satisfying Kraft inequality is identical to the optimal codeword length selection in the prefix coding in the context of information theory. This problem is well studied and the optimal strategy is known to yield the Shannon entropy as the average codeword length [2]. Interpreted in the context of KID assignment, average number of keys per member is equal to the entropy of the member revocation event. Theorem below summarizes the main results without the proof. Proof is standard in information theory and can be found in chapter 5 of [2].

Theorem 2. For a key assignment satisfying Kraft inequality, optimal average number of keys, excluding the root key and the SK, held by a member is given by the d-ary entropy $H_d = -\sum_{i=1}^{i=N} p_i \log_D p_i$ of the member revocation event. For a member i with probability of revocation p_i , satisfying the optimization criteria, the optimal number of keys l_i , excluding the root key and the SK, is given by

$$l_i = -\log_D p_i. (3)$$

The following properties which are important for member revocation and grouping of valid members to find minimal number of keys for transmission are summarized in the form of the lemma below. These are also part of standard information theory results, and are valid for the "codewords" formed by the concatenation of the keys as well:

Lemma 2.

- 1. Given two members of the group, the member with higher probability of revocation should be assigned larger number of keys. If $p_i > p_j$, then $l_i (= -\log_D p_i) > l_j (= -\log_D p_j)$.
- 2. There must be at least two members with the largest number of keys.
- 3. The largest number of keys held differ only by one key and these two sets correspond to the members with the least probabilities of revocation.
- 4. The average number of keys held by a member is never less than the entropy H_D of the member revocation event. It is equal to the entropy of the member revocation event iff $p_i = D^{-i}$. i.e. the probabilities are D-adic.

³ Member addition may change some keys but does not necessarily force the change of old keys other than the traffic encrypting key.

Sketch of the Proofs:

- 1. If $p_i > p_j$, log being a monotone function, $\log_d p_i > \log_D p_j$. Hence $-\log_D p_i < -\log_D p_j$, leading to $l_i (= -\log_D p_i) < l_j (= -\log_D p_j)$.
- 2. If there are no two members with the largest codeword, then we can reduce the largest codeword by at least one more bit and still ensure that all members have unique codeword assigned. This will violate the proof of optimality of the individual codeword lengths.
- 3. Results follow from the fact that the optimal value of the average number of keys held by a member is a minima with the value equal to the entropy of the member revocation event.

From these statements, we note that a member with higher probability of being revoked should be given fewer number of keys. Since the number of nodes along the path connecting the leaf and the root of the logical tree represents the number of keys held by the member, the member with higher probability is closer to the root of the logical tree. The following observation summarizes the nature of the rooted key key distribution architectures in [18, 19, 21, 20].

Among all the efficient rooted tree based key revocation strategies that satisfy Kraft inequality, results in [18, 19, 21, 20] have the maximum entropy, and hence corresponds to the maximum average number of keys held by a member for a given tree size.

Upper Bounds on the Integer Values of keys Since the optimal number of keys held by a member i with probability of revocation p_i which is given by $l_i = -\log_D p_i$ needs to be an integer, we need to compute the exact bounds on the total number of keys (including the root key and the traffic key), assuming that the probability of member revocations can be ordered in an ascending order at the time of key assignment. This value is given by the following theorem.

Theorem 3. The optimal average number of keys held by a member satisfies $H_D + 2 \le \hat{l} + 2 \le H_D + 3$.

Proof: Result showing $H_D \leq \hat{l} < H_D + 1$ is standard [2] and is not repeated here. Adding 2 to makeup for the total number of keys $H_D + 2$) yields the desired result $H_D + 2 \leq \hat{l} + 2 < H_D + 3$.

Since the average number of keys per member is $(\hat{l}+2)$, we note that the *optimal* number of average keys per member is at most 3 D-ary digits more than, and is at least 2 D-ary digits more than the *entropy of the member revocation* event.

4 Characterization of Collusion in Schemes [20, 21] Using Optimal Source Coding

Noting that we used member revocation probabilities and derived the optimal rooted-tree based key revocation schemes that eliminate redundancies in [18,

19, 21, 20], one may be tempted to conclude it may be appropriate to use the deterministic optimal coding techniques like the Huffman coding to develop a one-to-one map between the members and the keys assigned to them. Since the optimal number of keys led to rooted trees often called the Huffman trees, choosing the codes based on Huffman coding appears attractive from the point of using minimal number of individual keys to construct codewords.

We assert claim that using the Huffman coding is not the right approach when collusion needs to be avoided. To justify/make our point, we will first review the key assignment methods discussed in [20, 21] and then show that the results presented in [20, 21] for binary trees (a) fall under the category of the optimal Huffman coding and (b) provide one of the lowest possible integrity levels for member collusion.

The authors in [20] noted that given the binary index of a member, each bit in the index takes two values, namely 0 or 1. To follow the example given in [20], when N=8, $\log_2 8=3$ bits are needed to uniquely index all 8 members. The authors then proceeded to claim that since each bit takes two values, it can be symbolically mapped to a distinct pairs of keys. The table below reproduces the mapping between the ID bit # and the key mapping for the case in [20] for N=8:

ID Bit #0		
ID Bit #1		
ID Bit #2	K_{20}	K_{21}

where, the key pair (K_{i0}, K_{i1}) symbolically represents the two possible values of the ith bit of the member index. Although this table does provides a one-to-one mapping between the set of keys and the member index using only eight keys, the problem with this approach becomes clear if we map the table to the rooted tree structure. Figure 2 shows the mapping of the keys on the tree. (For the sake of clarity, not all the keys corresponding to the leafs are shown in figure 2). Adjacent leafs have K_{30}, K_{31} as the keys and this pair is repeated across the level. In fact, at any depth only two specific keys have been used and duplicated across the depth. If members corresponding to leafs 1, and 8 are compromised or to collude, entire set of eight keys will be exposed by these two members, i.e., this system will be *completely* broken independent of the size N of the group. Hence, this scheme is ranked very low in providing guarantees of privacy or integrity against collusion.

There are three different ways to interpret the collusion problems with approaches in [20, 21] based on rooted trees. We present them in the order of generality:

4.1 Interpretation Based on Minimal Number of Key Requirements

A simple way to interpret the shortcomings of results in [20, 21] is to note that $2 \log_2 N < N, \forall N > 4$. In order to prevent member collusion from being able to break the rest of the system, there must be at least N keys so that each member

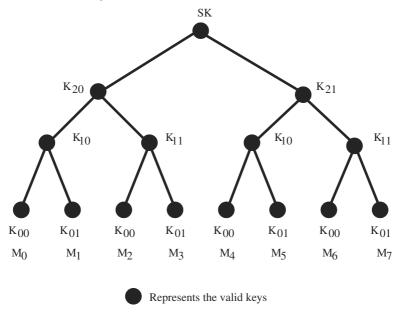


Fig. 2. The Logical Key Tree of [21, 20]

has a unique key and can be contacted at the time of member revocation. since $2\log_2 N < N \ (N>4)$ is the number of distinct keys used by the variation of rooted tree presented in [20, 21], and can be completely or partially compromised depending on the colluding members.

4.2 Interpretation based on Source Coding

For simplicity, we assume that the group size N is a dyadic number. Since we showed that the traditional binary rooted tree based rooted-tree [18, 19, 21, 20 leads to the maximum entropy of the member revocation event, the number of keys per member, $\log_2 N$, is same as the average number of keys per member. Also, the member indices each need $\log_2 N$ bits. The scheme in [20, 21] used a unique pair of keys to symbolically map each of bit positions of the the member index. Hence, a total of $2\log_2 N$ keys are used to uniquely represent each member index. This selection of keys can create a set of N unique indices and the codewords generated by concatenating $\log_2 N$ keys satisfy the Kraft inequality. Hence, this mapping of a unique pair of keys to each bit location corresponds to performing a Huffman coding with $2H_2(U)$ distinct keys, where $H_2(U) = \log_2 N$. If we use the notation (k_i, k_i) to denote the unique key pair representing the two possible binary values taken by the jth bit, we note that the collusion or compromise of two members holding keys k_j and k_j respectively will compromise the integrity of the key pair (k_j, \hat{k}_j) . The following lemmas summarize our observations:

Lemma 3. If the binary rooted key tree uses the optimal Huffman Coding for assigning members a set of keys based on $2\log_2 N$ (N>4) (here N is dyadic) distinct keys as in [20, 21], the whole system can be broken if any two members whose "codewords" or KIDs (NOT UIDs as in many other recent papers) are one's complement of each other collude or are compromised. Hence, the integrity systems in [20, 21] do not scale beyond 4 members in the presence of colluding members.

In a D-ary tree, each digit takes D values and the sum of these values is given by $\frac{D(D-1)}{2}$. Hence, if a set of k $(k \geq D)$ members whose ith bit values when summed lead to $\frac{D(D-1)}{2}$ collude, they will be able to fully compromise the ith bit location. This result is summarized by:

Lemma 4. For a D-ary tree with N members, the key corresponding to bit location b will be compromised by a subset of k ($k \geq D$) members whose symbolic value of the bit location b denoted by the set $\{b_1, b_2, \cdots, b_k\}$ satisfy $b_1 + b_2 \cdots b_k \equiv 0 \mod \frac{D(D-1)}{2}$.

4.3 Interpretation Based on Complementary Variables

The third interpretation is based on the notion of sets and includes a larger definition of collusion discussed under the category of complementary variables in [18]. The approach in [20, 21] is a special case of the complementary variable approach. If the secure group membership is a set such that every member is denoted by a unique key and that key is given to all other members but the member itself, at the time the member is to be revoked, all other members can use the key denoting the revoked member as the new key. For a set of N members, all the members will have (N-1) keys that corresponding to other members and no member will have the key denoting itself. Clearly, if two members collude, between them they will have all the future keys of the group. Hence, this kind of key assignment does not scale beyond 2 members.

4.4 Appropriate Mapping of the Keys to the Members

The main problem with the approach presented in [20, 21] is that the direct mapping of the values taken by each bit in the "codeword" to a unique pair of keys. As mentioned earlier, given a bit location h from the root of the tree, $2^{h-1}2^{h-1}=2^{2h-2}$ possible combinations of members can collude to compromise the keys corresponding to that bit. In order to avoid such repeated assignment of keys, only one internal node or bit location at a given depth should be assigned a particular key. This will guarantee that only the descendant leafs of that node will have access to the specific key assigned to that node. Hence, when a member is revoked, specific key will not be part of any "unrevoked" path.

5 Entropy Based Bounds on Average Key Generation Requirements and Conditions for Clustering

We showed that on average, at the time of member revocation $(2 + H_D)$ keys need to be updated. If each key is L bits long, then the average number of bits that need to be generated by the hardware after key revocation is $L(2 + H_D)$ bits. Since $H_D \leq \log_D N$ with equality attained iff all the members have equal revocation probabilities, the hardware need to be able to generate a worst case average of $L(2 + \log_D N)$ bits within the next unit of time of update to let the session continue.

Theorem 4. For a binary rooted tree based rooted-tree family of systems with keys of length L bits, the average number of bits B that need to be generated by the hardware at the time of member revocation, should satisfy $B \ge L(\log_D N + 2)$, with the average lower bound being attained *iff* all the members have equal probability of being revoked.

Proof: As shown earlier, average number of keys to be generated in the event of member revocation is given by $(2 + H_D) = 2 + \sum_{i=1}^{i=N} p_i l_i$. Hence, the hardware should be able to generate a total of $L(H_D+1)$ bits of suitable quality⁴ in unit of time to let the session continue without delays in the average sense. Desired lower bound follows from the observation that $H_D \leq H_D(U) = \log_D N$, with equality iff all the members have the same revocation probabilities.

From the above given theorem, if membership is too large for a single hardware to support the key generation, there need to be at least $\lceil \frac{L(H_D+2)}{B} \rceil$ units of hardware with capability of generating B bits in a unit of time. This result can also be interpreted from the point of view of splitting a group into clusters. If a group center can update only B ($L < B < L(2+H_D)$) bits in a unit of time, it may be appropriate to split the group center into a panel consisting of at least $\lceil \frac{L(H_D+2)}{B} \rceil$ centers each of which can update B ($L < B < L(2+H_D)$) bits in a unit of time.

6 Conclusions and Future Work

This paper showed that several properties of the recently proposed [18, 19, 20, 21] rooted tree based secure multicast key management schemes can be systematically studied using information theoretic concepts. By using the member revocation event as the basis of our formulation, we showed that the optimal number of average keys per member is related to the *entropy* of the member revocation event. We then proved that the currently available known rooted tree based strategies [18, 19, 20, 21] yield the maximum entropy among all the rooted tree based strategies and hence opt for the maximal average number of keys per member regardless of the values of the revocation probabilities. Using the optimal source coding strategy, we identified the *collusion* problem in [20, 21] resulting

⁴ Based on the application specific use of the key.

from performing the Huffman coding with $D\log_D N$ symbols. We also showed which subset of members need to collude or be compromised to break schemes such as the ones in [20, 21], regardless of the size of N. We showed that for a group with uniform revocation probabilities and using a binary tree, it is enough for two members with complementary keys to collude to break the scheme. We then showed that using the entropy of the member revocation event, we can set a bound for the minimal average case hardware key generation requirements. We also provided a simple rule for deciding group size based on hardware availability (or the number of hardwares required to support a size N, on average).

Acknowledgments

We would like to thank A. Ephremides and Jim Massey for early references on the rooted-tree algorithms by Massey. Relating key length to the entropy was due to observation by A. Ephremides that the keys are not shown to be related to classical information theroy. We would like of thank Benny Pinkas for providing reference [1].

References

- R. Canetti, T. Malkin, and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption", In Eurocrypt 99, pp. 456 - 470.
- T. Cover, J. Thomas, Elements of Information Theory, John Wiley & Sons, Inc, NY, 1991.
- 3. R. Gallager, Information theory and reliable communication, Wiley, NY, 1968.
- 4. J. L. Massey, "An Information-Theoretic Approach to Algorithms", Impact of Processing Techniques in Communications, In NATO Advanced Study Institutes Series E91, pp. 3-20, 1985.
- J. L. Massey, "Some Applications of Source Coding to Cryptography", In European Trans. on Telecom., Vol. 5, pp. 421-429, July-August 1994.
- H. N. Jendal, Y. J. B. Khun, and J. L. Massey, "An Information-Theoretic Approach to Homomorphic Substitution", In Advances in Cryptology-Eurocrypt'89, LNCS-434, pp. 382-394, 1990.
- 7. U. M. Maurer, "Secret Key Agreement by Public Discussion from Common Information", In IEEE Trans. IT, Vol 39, No. 3, 1993, pp 733-742.
- 8. R. Canetti, and B. Pinkas, "A taxonomy of multicast security issues", *Internet draft*, April 1999.
- Y. Desmedt, Y. Frankel, and M. Yung, "Multi-receiver/Multi-sender network security: efficient authenticated multicast feedback", *IEEE Infocom'92*, pp. 2045-2054.
- M. steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication", 3rd ACM Conf. on Computer and Communications Security", 1996.
- R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast Security: A Taxonomy and Efficient Reconstructions", Proceedings of IEEE Infocom'99.

- 12. D. A. McGrew and A. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", *Manuscript*, 1998.
- 13. A. Fiat and M. Naor, "Broadcast Encryption", Advances in Cryptology- Crypto'92, Lecture Notes in Computer Science. vol. 773, pp. 481-491, Springer-Verlag, Berlin Germany, 1993.
- A. Menezes, P. van Oorschot, and A. Vanstone, "Handbook of Applied Cryptography", CRC Press, Boca Raton, 1997.
- M. Naor and O. Reingold, "From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs", Advances in Cryptology-Crypto'98, Lecture Notes in Computer Science. vol. 1462, pp. 267-282, Springer-Verlag, Berlin Germany, 1998.
- $16.\ \ M.\ Luby,\ Pseudo-Random\ Functions\ and\ Applications,\ Princeton\ University\ Press,\\ 1996.$
- M. Brumester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System", Advances in Cryptology- Eurocrypt'94, Lecture Notes in Computer Science. vol. 950, pp. 275-286, Springer-Verlag, Berlin Germany, 1994.
- 18. D. M. Wallner, E. C. Harder, and R. C. Agee, "Key Management for Multicast: Issues and Architectures", Internet Draft, September 1998.
- C. K. Wong, M. Gouda, S. S. Lam, "Secure Group Communications Using Key Graphs", In *Proceedings of ACM SIGCOMM'98*, September 2-4, Vancouver, Canada.
- G. Caronni, M. Waldvogel, D. Sun, and B. Plattner, "Efficient Security for Large and Dynamic Groups", In Proc. of the Seventh Workshop on Enabling Technologies, IEEE Computer Society Press, 1998.
- 21. I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, "Key Management for Secure Internet Multicast Using Boolean Function Minimization Techniques", To apper in Proceedings of IEEE Infocom'99.
- S. Mittra, "Iolus: A framework for Scalable Secure Multicasting", In Proceedings of ACM SIGGCOM'97, pages 277–288, September 1997.
- H. Harney and C. Muckenhirn, "GKMP Architecture", Request for Comments (RFC) 2093, July 1997.
- R. Canetti, P-C. Cheng, D. Pendarakis, J. R. Rao, P. Rohatgi, D. Saha, "An Architecture for Secure Internet Multicast", Internet Draft, November 1998.
- T. Hardjono, B. Cain, and N. Doraswamy, "A Framework for Group Key Management for Multicast Security", Internet draft, July 1998.
- B. Quinn, "IP Multicast Applications: Challenges and Solutions", Internet draft, November 1998.
- H. Harney and C. Muckenhirn. "GKMP Specification". Internet RFC 2094, July 1997.
- A. Ballardie. "Scalable Multicast Key Distribution". Internet RFC 1949, May 1996.

Author Index

Abe, Masayuki130	Krovetz, Ted	216
An, Jee Hea252	Kumar, Ravi	609
Aumann, Yonatan65		
,	Lam, Kwok-Yan	315
Baras, John S624		
Beimel, Amos	MacKenzie, Philip	110
Bellare, Mihir	Maitra, Subhamoy	ر ۱۱ ۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰
	Mallan Tal	190
Black, John216	Malkin, Tal	
Boneh, Dan	Maurer, Ueli	
Boyko, Victor503	Micali, Silvio	
	Michels, Markus	
Camenisch, Jan413	Miner, Sara K	431
Canetti, Ran98		
Chari, Suresh398	Naccache, David	
Coppersmith, Don305	Naor, Moni	573
Coron, Jean-Sébastien1	Nguyen, Phong	
Coron, vour Socusion	1 (8 4) 644, 1 11048	
Dai Zana Dua	Okamoto, Tatsuaki	537
Dai, Zong-Duo315	Ostrovsky, Rafail	105
Di Crescenzo, Giovanni485	Ostrovsky, Karan	463
Durfee, Glenn	D . 1 G	22.4
	Patel, Sarvar	
Etzel, Mark234	Pinkas, Benny	
	Poovendran, Radha	624
Fiat, Amos		
Franklin, Matthew	Rabin, Michael O	65
Fujisaki, Eiichiro	Rabin, Tal	98
rujisaki, Elicilito	Rajagopalan, Sridhar	
	Rao, Josyula R	
Gafni, Eli372	Ramzan, Zulfikar	
Garay, Juan A449	Robshaw, M.J.B	
Gennaro, Rosario98		
Gilboa, Niv	Rogaway, Phillip	
Goldreich, Oded	Rohatgi, Pankaj	398
,		
Halevi, Shai216	Sahai, Amit	
Howgrave-Graham, Nick326	Sander, Tomas	
Howgrave-Granam, Nick520	Sarkar, Palash	
	Schoenmakers, Berry	148
Ishai, Yuval591	Shamir, Adi	19
	Staddon, Jessica	
Jaffe, Joshua388	Stern, Jacques	
Jakobsson, Markus449	Stern, Julien P	
Jarecki, Stanislaw98	Stern, Junear I	
Johansson, Thomas181	Ta-Shma, Amnon	555
Jönsson, Fredrik		
Jun, Benjamin388	Tassa, Tamir	334
Jutla, Charanjit S		
Julia, Charanju S598	Vadhan, Salil	467
Kipnis, Aviad19	Wagner, David	165
Knudsen, Lars R		
Kocher, Paul388	Ye, Ding-Feng	315
Krawczyk, Hugo98, 216, 270	Yin, Yiqun Lisa	372
,	, 1	