Ed Dawson   Serge Vaudenay (Eds.)

Progress in Cryptology – Mycrypt 2005

First International Conference
on Cryptology in Malaysia
Kuala Lumpur, Malaysia, September 28-30, 2005
Proceedings

Springer

Volume Editors

Ed Dawson
Queensland University of Technology
Information Security Institute
GPO Box 2434 (Level 7, 126 Margaret Street)
Brisbane Qld 4001, Australia
E-mail: e.dawson@qut.edu.au

Serge Vaudenay
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Security and Cryptography Laboratory (LASEC)
1015 Lausanne, Switzerland
E-mail: serge.vaudenay@epfl.ch

Library of Congress Control Number: 2005932807

CR Subject Classification (1998): E.3, D.4.6, K.6.5, C.2, K.4, F.2.1-2

ISSN	0302-9743
ISBN-10	3-540-28938-0 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-28938-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper   SPIN: 11554868   06/3142   5 4 3 2 1 0

Preface

Mycrypt 2005 was the inaugural international conference on cryptology hosted in Malaysia. The conference was co-organized by the Information Security Research Lab at Swinburne University of Technology (Sarawak Campus), NISER (National ICT Security and Emergency Response Centre) and INSPEM (Institute for Mathematical Research) at UPM (University Putra Malaysia). Mycrypt 2005 was held in Kuala Lumpur, Malaysia during September 28–30 2005, in conjunction with the e-Secure Malaysia 2005 convention.

There were 90 paper submissions from 23 countries covering all areas of cryptologic research, from which 19 were accepted. We would like to extend our thanks to all authors who submitted papers to Mycrypt 2005. Each paper was sent anonymously to at least 3 members of the International Program Committee for reviews and comments. The review comments were then followed by discussions among the Program Committee. A recipient of the Best Paper Award was also selected after voting among Program Committee members. The winning paper was "Distinguishing Attacks on T-functions" by Simon Künzli (FH Aargau, Swizerland), Pascal Junod (Nagravision SA, Switzerland) and Willi Meier (FH Aargau, Swizerland). These proceedings contain revised versions of all the accepted papers.

The conference program included three keynote papers: Hideki Imai (Tokyo University) presented a paper entitled "Trends and Challenges for Securer Cryptography in Practice". Moti Yung (Columbia University) presented a paper entitled "Efficient Secure Group Signatures with Dynamic Joins and Keeping Anonymity Against Group Managers". Colin Boyd (QUT) presented a paper entitled "Security of Two-Party Identity-Based Key Agreement".

We are extremely grateful for the time and effort of all the members of the Program Committee in the review process. Their names may be found overleaf. This group was assisted by an even larger group of experts. A list of names is also provided. We hope that it is complete. We give special thanks to Thomas Baignères and Matthieu Finiasz for handling the servers during the review process and preparing the proceedings. The Web-based submission software was written by Chanathip Namprempre with additions by Andre Adelsbach and Andrew Clark. The review process and Program Committee discussions were supported by the Web-based review software developed by Bart Preneel, Wim Moreau and Joris Claessens.

We wish to acknowledge the excellent Mycrypt Local Organizing Committee led by the General Chair, Raphael C.-W. Phan; and the support of MOSTI (Ministry of Science, Technology & Innovation), MEWC (Ministry of Energy, Water & Communications), MCMC (Malaysia Communications & Multimedia Commission), MAMPU (Malaysian Administrative Modernisation & Management Planning Unit), SIRIM (Standards & Industrial Research Institute of Malaysia) and the Malaysian National Computer Confederation (MNCC).

September 2005 Ed Dawson and Serge Vaudenay

Mycrypt 05

International Conference on Cryptology in Malaysia

September 28–30, 2005, Kuala Lumpur, Malaysia

General Chair

Raphael C.-W. Phan, *Swinburne University of Technology Sarawak, Malaysia*

Program Chairs

Ed Dawson, *Queensland University of Technology (QUT) Brisbane, Australia*

Serge Vaudenay, *Ecole Polytechnique Fédérale de Lausanne Lausanne, Switzerland*

Program Committee

Feng Bao	Institute for Infocomm Research, Singapore
Jean-Sébastien Coron	University of Luxembourg, Luxembourg
Ronald Cramer	Leiden University, The Netherlends
Ed Dawson	Queensland University of Technology, Australia
Yvo Desmedt	University College London, UK
Juan M. González Nieto	Queensland University of Technology, Australia
Helena Handschuh	Gemplus, France
Norbik Idris	Universiti Technologi Malaysia, Malaysia
Antoine Joux	DGA and Université Versailles St. Quentin, France
Marc Joye	Gemplus & CIM-PACA, France
Jonathan Katz	University of Maryland, USA
Kwangjo Kim	Information & Communications University, Korea
Xuejia Lai	Shanghai Jiaotong University, China
Kwok-Yan Lam	Tsinghua University, China
Arjen K. Lenstra	Lucent Technologies, USA and Technische Universiteit Eindhoven, The Netherlands
Stefan Lucks	University of Mannheim, Germany
Wenbo Mao	Hewlett-Packard Labs, UK
Mitsuru Matsui	Mitsubishi Electric, Japan
Rushdan Md Said	University Putra Malaysia, Malaysia
Chris Mitchell	Royal Holloway, University of London, UK
Shiho Moriai	Sony Computer Entertainment, Japan
Gregory Neven	Katholieke Universiteit Leuven, Belgium
Phong Nguyen	CNRS/Ecole Normale Superieure, France
Andrew Odlyzko	University of Minnesota, USA

Eiji Okamoto	University of Tsukuba, Japan
Tatsuaki Okamoto	NTT, Japan
Raphael C.-W. Phan	Swinburne University of Tech., Sarawak, Malaysia
Josef Pieprzyk	University of Macquarie, Australia
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Jean-Jacques Quisquater	Université Catholique de Louvain, Belgium
Pandu Rangan	IIT Madras, India
Vincent Rijmen	Graz University of Technology, Austria
Mohammad Umar Siddiqi	Multimedia University, Malaysia
Serge Vaudenay	EPFL, Switzerland
Sung-Ming Yen	National Central University, Taiwan

External Referees

Michel Abdalla
Masayuki Abe
Kazumaro Aoki
Frederik Armknecht
Gildas Avoine
Thomas Baignères
Chris Vanden Berghe
Olivier Billet
Wieb Bosma
Colin Boyd
An Braeken
Christophe De Cannire
Dario Catalano
Julien Cathalo
Chien-ning Chen
Benoît Chevallier-Mames
Kuo-Zhe Chiou
JaeGwi Choi
Andrew Clark
Scott Contini
Nicolas Courtois
Christophe Doche
Serge Fehr
Matthieu Finiasz
Soichi Furuya
Praveen Gauravaram
Damien Giry
Eu-Jin Goh
Bok-Min Goi
Louis Granboulan
Robbert de Haan
Chao-Chih Hsu
Tetsu Iwata
Ellen Jochemsz
Pascal Junod
Seny Kamara
Hyun Jeong Kim
Kazukuni Kobara
Caroline Kudla
Ulrich Khn
Tanja Lange
Joe Lano
Philippe Léglise
Julie Lescut
Benoit Libert
Vo Duc Liem
Hsi-Chung Lin
Pascal Manet
Stefan Mangard
Bill Millan
Atsuko Miyaji
Havard Molland
Jean Monnerat
Peter Montgomery
Sumio Morioka
Siguna Mueller
Hirofumi Muratani
Jorge Nakahara, Jr.
Kenny Nguyen
Svetla Nikova
Elisabeth Oswald
Pascal Paillier
Olivier Pereira
Duong Hieu Phan
Christian Rechberger
Leonid Reyzin
Yasuyuki Sakai
Palash Sarkar
Taizo Shirai
Joseph Silverman
Jason Smith
Martjn Stam
François-Xavier Standaert
Dirk Stegemann
Ron Steinfeld
Hung-Min Sun
Katsuyuki Takashima
Qiang Tang
Emin Tatli
Shinobu Ushirozawa
Eric Verheul
Zhiguo Wan
Guilin Wang
Huaxiong Wang
Shiuh-Jeng Wang
Benne de Weger
Andreas Wespi
William Whyte
Christopher Wolf
Chi-Dian Wu
Yongdong Wu
Yanjiang Yang
Bo Zhu
Huafei Zhu

Table of Contents

Trends and Challenges for Securer Cryptography in Practice

Hideki Imai

Institute of Industrial Science, The University of Tokyo,
Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology

As the importance of information security is widely recognized today, development of cryptography in practical use is rapidly taking place. On the other hand, however, many cases have been reported, where problems are found in the cryptographic systems already in use, or where the cryptographic systems are broken. Causes for a cryptographic system to get corrupted can be: defects in cryptographic algorithm designs; defects in implementation; defects in attack models and definitions of security; progress in computers and attack algorithms; inapplicability due to the change of environment. It is to be noted that the cryptographic system that has been created in the circumstance where there can be some kind of defects is generally vulnerable to breakdown. In the world of cryptography, we should regard "Anything that can happen, happens."

In the first half of my lecture, I will show you some examples of vulnerability in cryptography and measures against it. First, as examples of imminent crises, I will talk about the WEP (Wired Equivalent Protocol) vulnerability, which is a wireless LAN encryption; vulnerability in hash functions; implementation attacks in IC cards; and the issue of "quantum cryptography" Y-00. Next, as a near-future danger, I will talk about serious attacks to a 1024-bit RSA cryptosystem, and lastly, the realization of quantum computers, as an example of future crisis. It is important to evaluate cryptographic systems systematically and continually, in dealing with these crises. As an example of an organization that performs such execution, I will introduce CRYPTREC of Japan.

However, even if all these dangers are prevented, cryptographic systems can be broken, because we cannot prevent human errors and unauthorized acts inside the cryptographic systems completely. In the second part of this lecture, I will talk about Fail-Safe Techniques for Information Security, as a measure against such human-related issues. First, I will discuss the fundamental concept of these techniques. Next, I will introduce, as examples of the fundamental technology in this field, how to construct public-key infrastructures and person authentication systems that are robust against the leakage of secret keys and secret verification data.

Lastly, I will introduce the Research Center for Information Security, at the National Institute of Advanced Industrial Science and Technology, which was established on April 1st of this year and is expected to contribute greatly, in future, in constructing and maintaining the high-level information security of IT systems.

Distinguishing Attacks on T-Functions

Simon Künzli[1], Pascal Junod[2], and Willi Meier[1]

[1] FH Aargau, 5210 Windisch, Switzerland
{s.kuenzli, w.meier}@fh-aargau.ch
[2] Nagravision SA (Kudelski Group), 1033 Cheseaux, Switzerland
pascal.junod@nagra.com

Abstract. Klimov and Shamir proposed a new class of simple cryptographic primitives named T-functions. For two concrete proposals based on the squaring operation, a single word T-function and a previously unbroken multi-word T-function with a 256-bit state, we describe an efficient distinguishing attack having a 2^{32} data complexity. Furthermore, Hong *et al.* recently proposed two fully specified stream ciphers, consisting of multi-word T-functions with 128-bit states and filtering functions. We describe distinguishing attacks having a 2^{22} and a 2^{34} data complexity, respectively. The attacks have been implemented.

Keywords: Stream cipher, T-function, square mapping, distinguishing attack, statistical cryptanalysis

1 Introduction

Binary additive stream ciphers encrypt a plaintext stream by combining it with a key stream by means of an XOR operation (the decryption simply being the XOR of the key stream with the ciphertext stream). The key stream consists of a pseudo-random bit sequence usually generated by iteration of an *update function*, the latter being initialized with a secret state. One expects that the sequence generated by a cryptographically secure stream cipher is statistically indistinguishable from a truly random sequence (and this for any adversary with some limited computational power), and that there exists no key-recovery attack better than brute-force.

Recently, Klimov and Shamir [7, 8, 9, 10, 6] proposed a new framework for highly efficient mappings which could be used as primitives in stream ciphers and other cryptographic schemes. These primitives consist of *triangular functions* (T-functions) which are built with help of fast arithmetic and Boolean operations widely available on high-end microprocessors or on dedicated hardware implementations; these mappings come with provable properties such as invertibility and a single-cycle structure. As an example, the mapping TF-0 is proposed in [7], which is defined by $x \mapsto x + (x^2 \vee C) \bmod 2^n$ for an n-bit state x and with $C \equiv 5, 7 \pmod 8$. As the maximal length of a cycle may be too short for typical values of n (e.g. $n = 64$), and as state-recovery attacks have been described [2, 8], TF-0 is not meant to be directly used for cryptographic purposes.

Considering cryptographic applications, several efficient multi-word T-functions are proposed in [9]. Some of these proposals have been broken by Mitra and Sarkar [13] using time-memory tradeoffs. Based on the results of Klimov and Shamir, a new class of multi-word T-functions and two fully specified stream ciphers have been proposed by Hong *et al.* [3, 4]. Their schemes TSC-1 and TSC-2 have a transparent design and allow for some flexibility.

1.1 Contributions of This Paper

In this paper, we analyse several proposals of T-functions and exhibit substantial weaknesses in some of these constructions. The flaws are extended to dedicated attacks.

First we analyse the statistical properties of the pure square mapping, which allows us to find an efficient distinguisher (with an expected 2^{32} data complexity) on TF-0 as well as on a previously unbroken multi-word mapping described in [9] and labeled here as TF-0m, both based on the squaring operation. TF-0m operates on a 256-bit state and the output sequence consists of the 32 most significant bits.

Then, we cryptanalyse the TSC-family of stream ciphers [4], which operates on a 128-bit state and outputs 32 bits of the state using a filtering function. We find a very efficient distinguisher for TSC-1 with an expected 2^{22} data complexity; for TSC-2, we describe a different distinguishing attack with an expected 2^{34} data complexity.

To confirm our theoretical results, the distinguishing attacks have been implemented and run many times with success. Our distinguishers have a negligible error probability and a remarkably small time complexity.

1.2 Notational Conventions

We analyse cryptographic schemes consisting of an internal state $x \in \mathcal{X}$, an update function $\mathsf{f} : \mathcal{X} \to \mathcal{X}$ and an output function $\mathsf{g} : \mathcal{X} \to \mathcal{Y}$. In the case where time instants are relevant, we will denote x^t the state at time t (distinction of powers will be clear from the context). Hence, the iterative scheme maps the state x^t to $x^{t+1} = \mathsf{f}(x^t)$ and outputs $y^t = \mathsf{g}(x^t)$. The seed of the iteration is obtained from the secret key with help of a key scheduling process. The keystream K consists in the concatenation of successive outputs, namely $K = y^0||y^1||\cdots$.

We assume throughout this paper the threat model of a known-plaintext attack, i.e., we assume to know some part of the keystream K. Our purpose is then to distinguish K from a uniformly distributed random sequence, or to recover the state at any time.

In the case where the state is a vector formed by some words, we will denote a single word by x_j and the state as $x = (x_0, x_1, \ldots)$. Adopting the common notation, $[x]_i$ is the $(i+1)$-st least significant bit-slice of the state, $[x]_0$ denoting the rightmost bit-slice. Consequently, $[x_j]_i$ is the $(i+1)$-st least significant bit of word j. The operation $\mathrm{msb}_m(x)$ states for the m most significant bits of x. Arithmetic operations are performed modulo 2^n with typical word size $n = 32$ or 64 bit. Boolean operations are performed on all n bits in parallel and are denoted by $\wedge$ (AND), $\vee$ (OR), and by $\oplus$ (XOR). Finally, $\lll k$ denotes a cyclic left shift by k positions.

2 Cryptanalysis of Square Mappings

Klimov and Shamir have proposed different types of T-functions based on the squaring operation [7, 9]. After introducing the framework of this section, we focus on the pure square mapping and derive a hypothesis about their probability distribution. This distribution is used in order to distinguish the proposed mappings TF-0 and TF-0m with significant advantage.

Let us consider a scheme which consists of an update function f and an output function g with the notation of Sect. 1.2. Let us further define the random variables X and X' over the set $\mathcal{X} = \{0,1\}^n$, with uniformly distributed X and with $X' = \mathsf{f}(X)$. Equivalently, Y and Y' are random variables over $\mathcal{Y} = \{0,1\}^m$ with uniformly distributed Y and with $Y' = \mathsf{g}(\mathsf{f}(X))$. Given $\Pr_Y$, $\Pr_{Y'}$ and some uniform random or pseudo-random output respectively, we can perform a statistical test (e.g. a Neyman-Pearson test, see Appendix A for more details) in order to assign the output to a distribution. We are interested in the overall complexity of the distinguisher corresponding to some designated overall error probability π_{e}.

For small[1] word sizes n, the distribution $\Pr_{Y'}$ can be determined by an exhaustive computation of $\mathsf{g}(\mathsf{f}(x))$ for all 2^n elements x, resulting in a precomputation time complexity of $\mathcal{O}(2^n)$ and a memory complexity (measured with the number of required memory cells) of $\mathcal{O}(2^m)$. Given both distributions and a designated overall error probability, the data complexity of an optimal distinguisher is estimated with help of the squared Euclidean imbalance (see Appendix A). We assume that the test is performed in real-time, hence we do not need additional memory in order to store the data. The online time complexity is identical to the data complexity.

However, a precomputation of $\Pr_{Y'}$ might be infeasible for large values of n (e.g. $n = 64$ bit). We perform some detailed analysis of $\Pr_{Y'}$ for small word sizes n and establish an analytical hypothesis for the approximated distribution of Y', considering *only the most biased* elements. This significantly reduces the offline time and memory complexity, but might increase the online time and data complexity of the distinguisher, given some π_{e}. For small word sizes n, the hypothesis can be verified with the accurate distributions, and for large n, the quality of the hypothesis will be directly examined by the experimental data complexity of the distinguisher.

2.1 Distribution of the Pure Square Mapping

Let us define the pure square mapping $\mathsf{f}(x) = x^2 \bmod 2^n$ and $\mathsf{g}(x) = \mathrm{msb}_m(x)$ with $m = n/2$, which we will refer as PSM. Apart from the least significant bit, f

[1] The term *small* is used with respect to current computational possibilities, i.e. $n \lesssim 40$ bit for personal computers nowadays.

is a T-function. Iteration produces some fixed points such as 0 or 1, hence f can not be considered as an update function for a real application. However, we will be able to reduce more complex single-cycle mappings to some modified square mappings and apply the results obtained in this section; in other words, we will consider the pure square mapping as an ideal case, resulting in distinguishers with minimal data complexity compared to modified square mappings.

We first mention that Klimov and Shamir [7] found an analytical expression for probabilities of single bits of the square mapping. Applying the notation $X' = \mathsf{f}(X)$ for an uniformly distributed X, they found that $\Pr([X']_0 = 0) = \frac{1}{2}$, $\Pr([X']_1 = 0) = 1$ and $\Pr([X']_i = 0) = \frac{1}{2}(1 + 2^{-\frac{i}{2}})$ for $i > 1$. However, as we will have to deal with an additional carry bit later on (which would reduce this bias significantly), we are more interested in the distribution of words.

We explain how to derive highly biased probability distributions for $X' = \mathsf{f}(X)$ and $Y' = \mathsf{g}(\mathsf{f}(X))$. As shown in the next proposition, f is not a permutation, resulting in an unbalanced distribution of X' (there are some predictable elements $\mathsf{f}(x)$ with exceptionally large bias).

Proposition 1. *Consider the function* $\mathsf{f} : \{0,1\}^n \to \{0,1\}^n$ *with* $\mathsf{f}(x) = x^2 \bmod 2^n$. *For successive elements* $x \in \{0, \ldots, 2^n - 1\}$*, the images* $\mathsf{f}(x)$ *have a cyclic structure with cycle length* 2^{n-2}*. Hence* f *is neither injective nor surjective.*

Proof. As $x^2 - \left(2^{n-1} + x\right)^2 = 0 \bmod 2^n$, we have two cycles of length 2^{n-1}, and as $\left(2^{n-2} + x\right)^2 - \left(2^{n-2} - x\right)^2 = 0 \mod 2^n$, both cycles have two mirrored sequences of length 2^{n-2}. Hence the output of successive numbers x has the shape $abc \ldots cbaabc \ldots cba$. □

Due to the specified output function in PSM, the bias is transferred to the distribution of Y'. For a truly random scheme, any element of the output occurs with probability $\pi_0 = 2^{-n/2}$. For the particular scheme PSM, we observed (for small word sizes n) that there exist 4 outcomes with biased probability $2 \cdot \pi_0$, 12 outcomes with biased probability $1.5 \cdot \pi_0$ and so on. This property appears to be independent of n, and we therefore can establish a hypothesis for the most biased elements (which are explicitly known). Let $\mathcal{Y}_i$ be the aggregate containing elements of constant biased probability π_i. The parameter s_i denotes the cardinality of $\mathcal{Y}_i$, and n_i denotes the minimal word size for a stable occurrence of π_i. The parameters n_i, s_i and π_i are summarized in Tab. 1. Then we have for $i = 0, \ldots, k$ (limited by the condition $n \geq n_k$)

$$\begin{aligned} \mathcal{Y}_0 &= \{2^{(n-n_0)/2} \cdot j^2; \qquad j = 0, \ldots, s_0\} \\ \mathcal{Y}_i &= \{2^{(n-n_i)/2} \cdot (1 + 8j); j = 0, \ldots, s_i\} \\ \mathcal{Y}_\infty &= \mathcal{Y} - \textstyle\sum \mathcal{Y}_i \ . \end{aligned} \tag{1}$$

The values in Tab. 1 are determined with empirical methods, however n_i and s_i are exact at least for word sizes within our computational possibilities. In the case of PSM, π_i is exact for $i = 0, 1$, but fluctuating for $i > 1$ so we have to take an average value. A further approximation is done with the remaining elements in $\mathcal{Y}_\infty$, which are assigned to a constant (standardised) probability. The number of aggregates k determines the accuracy of the approximation. However, k is constrained by the condition $n < n_k$, and as the values of π_i are only accurate for $n_i \approx 40$, we usually choose $k = 8$ for $n > 40$ bit. This corresponds to a memory complexity of 2^{17}. Regarding the complexities of a distinguisher, increasing the number of aggregates k is coupled with more time, more memory and less data.

Table 1. Parameters of the approximated distribution for the first 9 aggregates

i	0	1	2	3	4	5	6	7	8
$\pi_i 2^m$	2.000	1.500	1.200	1.100	1.050	1.030	1.002	1.005	1.003
$n_i 2^{-2}$	2	3	4	5	6	7	8	9	10
$\log_2(s_i)$	2	3	5	7	9	11	13	15	17

2.2 Attacking the Single-Word Mapping TF-0

Let us now consider the running single-word proposal TF-0 with the update function $\mathsf{f}(x) = x + (x^2 \vee C) \bmod 2^n$ where $C \equiv 5, 7 \pmod 8$, and with the output function $\mathsf{g}(x) = \text{msb}_m(x)$ where $1 \leq m \leq n/2$ as described in [7,10]. As the low-order bits are known to be weak, the authors of the scheme proposed $m = 1, 8, 16, 32$ for the standard word size $n = 64$ bit. Klimov and Shamir showed that f is an invertible T-function over an n-bit state x with a single cycle of length 2^n. The number of extracted bits m controls a tradeoff between security and efficiency of the scheme. We give some relationship to PSM with the next proposition.

Proposition 2. *Consider the scheme* TF-0*. If one requires $C < 2^{n-m}$, it is* $\mathsf{g}(\mathsf{f}(x)) - \mathsf{g}(x) = \mathsf{g}(x^2) + \alpha \mod 2^m$ *for $n - m > 2$ and for a carry bit $\alpha \in \{0, 1\}$.*

Proof. As $\mathsf{f}(x) = y = x + (x^2 \vee C) \bmod 2^n$, we conclude $y - x \equiv x^2 \vee C \pmod{2^n}$ for $C < 2^{n-m}$. Hence, $\mathsf{g}(y - x) \equiv \mathsf{g}(x^2 \vee C) \pmod{2^m}$ and $\mathsf{g}(y - x) \equiv \mathsf{g}(x^2) \pmod{2^m}$ for $C < 2^{n-m}$. We finally have $\mathsf{g}(y) - \mathsf{g}(x) - \alpha \equiv \mathsf{g}(x^2) \pmod{2^m}$ for $C < 2^{n-m}$ and for some carry bit $\alpha \in \{0, 1\}$. □

Proposition 2 states that the difference of two consecutive outputs of TF-0 differs only by an additive carry bit $\alpha \in \{0, 1\}$ from the output of PSM. Therefore, we may accurately approximate the distribution of the random variable $\mathsf{g}(\mathsf{f}(X)) - \mathsf{g}(X)$ by the distribution of the random variable Y' of PSM (i.e., we neglect the influence of the carry bit).

We choose standard parameters $C = 5$ and $m = n/2$. In order to perform a test for large values of n, we approximate the distribution $\Pr_{Y'}$ with the hypothesis described in Sect. 2.1, using an optimal number of aggregates. The data complexities are estimated according to (9) and verified with experiments. We got an experimental data complexity of 2^{32} for $n = 64$ bit, which turns out to be very close to the estimated value, and somewhat larger than the lower limit derived by extrapolation for the accurate probability distribution.

If the scheme is used as a pseudo-random number generator in large computer simulations, the output may not be considered as random after 2^{32} iterations, although we have a single-cycle of 2^{64} states. This observation is consistent with the practice nowadays, not to use more data than $\sqrt{P}$ of a pseudo-random number generator (PRNG) with period P. However, we also examined modified output functions with a smaller number of extracted bits m. Experiments show that (independently of the word size n), decreasing m by one bit increases the data complexity by a factor of 2. We conclude that, in contradiction to previous assumptions, not only the lower bits of this T-function are weak, but also the higher bits. This is an intrinsic property of the scheme, which will have consequences for other square mappings and may have consequences for more complicated output functions.

We mention that state-recovery attacks on TF-0 have been described in [2, 8]. Moreover, Mitra and Sarkar [13] described a time-memory tradeoff for the squaring problem, which may be applied to consecutive output differences of TF-0. The most efficient algorithms have a complexity of about 2^{16}.

2.3 Attacking the Multi-word Mapping TF-0m

Several multi-word update functions proposed in [9] have been attacked with a time-memory tradeoff by Mitra and Sarkar [13]. We now present a distinguishing attack against a multi-word proposal which has not been broken yet, and which we will refer as TF-0m. The update function f corresponds to (12) in [9], it is an invertible T-function over a $4n$-bit state $x = (x_0, x_1, x_2, x_3)$ with a single cycle of length 2^{4n}:

$$\mathsf{f} : \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 + (s_0^2 \vee C_0) \\ x_1 + (s_1^2 \vee C_1) + \kappa_0 \\ x_2 + (s_2^2 \vee C_2) + \kappa_1 \\ x_3 + (s_3^2 \vee C_3) + \kappa_2 \end{pmatrix} . \qquad (2)$$

It is $s_0 = x_0$, $s_1 = s_0 \oplus x_1$, $s_2 = s_1 + x_2$, $s_3 = s_2 \oplus x_3$. The constants are satisfying $[C_i]_0 = 1$ for $i \in \{0, 1, 2, 3\}$, and $[C_3]_2 = 1$. All operations are carried out on n bit words and κ_i denotes the carry bit of x_i. The output function is $\mathsf{g}(x) = \mathrm{msb}_m(x_3)$ with $m = n/2$. We choose the standard word size $n = 64$ bit.

The multi-word update function (2) consists of 4 approximatively independent and identically distributed (iid) random variables similar to the single-word update function of TF-0. We may concentrate only on the most significant variable x_3. The argument to be squared s_3 can be approximated as uniformly distributed, and therefore produces the same output as x^2. The carry bit modifies the output with a probability of 2^{-33}; this infrequent event will not have a significant influence to the distinguisher. Therefore, we do not have to modify the approximate distribution used for the distinguisher. Theoretical data complexity remains the same, and simulations result in an experimental data complexity of 2^{32} for a 256 bit state with 224 unknown bits. We have performed 20 experiments, observing no incorrect decision of our distinguisher. The data complexity is very close to the complexity for TF-0, confirming our assumption on the influence of κ and s.

We emphasize the practical applicability of this result and the small number of required data, compared to the large number of unknown bits. As before, we also considered to extract less bits $m < n/2$. Again, we found that decreasing m by one bit increases the data complexity by a factor of 2. Hence reduction of m may still not prevent practical attacks.

3 Cryptanalysis of TSC

We start this section with a description of the recent proposal of stream cipher family TSC [4]. We find a very efficient distinguishing attack on TSC-1, as well as a distinguishing attack on TSC-2.

3.1 Description of the Schemes

The hardware-oriented stream cipher family TSC consists of a state vector of 128 bits $x = (x_0, x_1, x_2, x_3)$, an update T-function f and an output function g. The update function consists of an odd 32-bit parameter $\alpha(x)$ and a single-cycle S-box S, mapping a 4 bit input to a 4 bit output. If $[\alpha]_i = 0$, then the mapping S^e is applied on bit-slice i of the state, otherwise the mapping S^o is applied. e (resp. o) is an even (resp. odd) number. This procedure is repeated for all 32 bit-slices in a single update period. With the satisfaction of these properties, f is a single-cycle T-function, hence the period of the cipher is 2^{128}.

The odd parameter is defined by $\alpha = (p + C) \oplus p \oplus 2s$ with a constant C, $p = x_0 \wedge x_1 \wedge x_2 \wedge x_3$ and $s = x_0 + x_1 + x_2 + x_3$. Except for the lower few bits, each output bit of α is equal to 1 almost half of the time. Due to the properties of an odd parameter, one has $[\alpha]_0 = 1$, meaning that the least significant bit-slice is always mapped by S^o. Consequently, the bits from the least significant bit-slice of the state will be referred as *irregular bits.*

Let us define the specified proposals. In TSC-1, the powers of the S-box are $e = 2$ and $o = 1$, the constant used in the odd parameter is $C =$ `0x12488421`, and the S-box (in standard notation) and the output function are defined by

$$\begin{aligned} \mathsf{S} &= (3,5,9,13,1,6,11,15,4,0,8,14,10,7,2,12) \\ \mathsf{g}(x) &= (x_{0 \lll 9} + x_1)_{\lll 15} + (x_{2 \lll 7} + x_3) \ . \end{aligned} \tag{3}$$

In TSC-2, one has $e = 0$ (hence, the identical mapping is used), $o = 1$ and $C =$ `0x00000001`. The S-box and the output function are defined by

$$\begin{aligned} \mathsf{S} &= (5,2,11,12,13,4,3,14,15,8,1,6,7,10,9,0) \\ \mathsf{g}(x) &= (x_{0 \lll 11} + x_1)_{\lll 14} + (x_{0 \lll 13} + x_2)_{\lll 22} + (x_{0 \lll 12} + x_3) \ . \end{aligned} \tag{4}$$

The output functions have a period of 2^{128}, however, three state variables in the output equation determine the remaining variable, hence the maximum security of the ciphers is 96 bit. Furthermore, there are some time-memory tradeoffs on TSC with large precomputation time complexities.

3.2 Attacking the Stream Cipher TSC-1

In this section, we present a linearisation attack on TSC-1. Probabilistic linear relations in the update function (i.e. relations between state bits at different time instants) and in the output function (i.e. relations between state bits and output bits) are combined, in order to obtain relations between output bits at different time instants. Provided that the relations are biased, the output of TSC-1 can be distinguished from a random stream.

Let us first discuss a linear approximation of the T-function. We focus on a single bit $[x_j^t]_i$ and analyse the statistical effect of $\varDelta$ iterations to this bit. Let $Y_\varDelta$ be the indicator variable of the event $[x_j^t]_i = [x_j^{t+\varDelta}]_i$, implying that a fixed bit is repeated after $\varDelta$ iterations. After $\varDelta$ iterations, bit-slice i (including the bit under observation) is mapped δ times by S, with $\varDelta \leq \delta \leq 2\varDelta$ (the mapping S is applied $2\varDelta - \delta$ times, and the mapping S^2 is applied $\delta - \varDelta$ times). Hence, in order to compute $\Pr(Y_\varDelta = 1)$, we have to analyse the distribution of δ and the bit-flip probabilities of the mappings S^δ.

Let us denote $b_\varDelta(\delta)$ the probability that after $\varDelta$ iterations, the S-box is applied δ times. For regular bit-slices, we reasonably assume equal probabilities for the application of S and S^2 (which is, however, a simplification for some lower bit-slices), and binomial distribution for $b_\varDelta$,

$$b_\varDelta(\delta) = \binom{\varDelta}{\delta - \varDelta} \cdot \left(\frac{1}{2}\right)^\varDelta . \tag{5}$$

For the irregular bit-slice, it is $b_\varDelta(\delta) = 1$ for $\delta = \varDelta$, and zero otherwise. In order to describe the effect of the mappings S^δ, let us analyse the S-box. We will denote w an uniform random number $0 \leq w \leq 15$, and i an index $0 \leq i \leq 31$. Let also X_δ be the indicator variable of the event $[w]_i = [\mathsf{S}^\delta(w)]_i$ for any fixed bit position i. The S-box is designed such that the *bit-flip probability* for an application of S and S^2 is balanced. However, there is a huge bias of the bit-flip probability for some multiple applications of S, namely for $\Pr(X_\delta = 1)$ with $\delta = 0 \mod 4$ (this observation is of course portable to the mapping S^2). We find $\Pr(X_4 = 1) = \Pr(X_{12} = 1) = 1/8$, $\Pr(X_8 = 1) = 3/4$ and of course $\Pr(X_{16} = 1) = 1$. These results are independent of bit-position i, other values of δ result in balanced probabilities.

Finally, the bit-flip probability $P(Y_\varDelta)$ of a single bit in the state for $\varDelta$ iterations simply becomes the weighted sum

$$\Pr(Y_\varDelta = 1) = \sum_{\delta=\varDelta}^{2\varDelta} \Pr(X_\delta = 1) \cdot b_\varDelta(\delta) . \tag{6}$$

We find a maximal bias for $\varDelta = 3$ with $\Pr(Y_3 = 1) = 0.3594$, and still a large bias for many other values of $\varDelta$. The predicted probabilities are in good agreements with experiments. In the case of irregular bits, (6) simply becomes $\Pr(Y_\varDelta = 1) = \Pr(X_\varDelta = 1)$ with a large bias for $\varDelta = 0 \mod 4$.

In the fictive case of a perfect single-cycle S-box (which, however, does not exist) with $\Pr(X_\delta = 1) = 1/2$ for $\delta \neq 16$ and $\Pr(X_{16} = 1) = 1$, (6) becomes

$\Pr(Y_\Delta = 1) = (b_\Delta(16) + 1)/2$ for regular bits. A maximal bias is obtained for $\Delta = 11$, resulting in $\Pr(Y_{11} = 1) = 0.6128$.

Let us combine the relation (6) with a simple linear approximation of the output function. The bias of Y_Δ strikes through the output function, such that the loops in the state are also present in the output. We consider a single bit $[y^t]_i$ of the output and analyse the statistical effect of Δ iterations to this bit. Let Z_Δ be the indicator variable of the event $[y^t]_i = [y^{t+\Delta}]_i$, implying that a fixed bit of the output is repeated after Δ iterations. We approximate the output function by $[y]_i = [x_0]_{i+8} \oplus [x_1]_{i+17} \oplus [x_2]_{i+25} \oplus [x_3]_i \oplus c$, for $i = 0, \ldots, 31$ (additions of indices are performed modulo 32) and a carry bit $c \in \{0, 1\}$. For bit-positions $i = 0, 7, 15, 24$, one irregular bit is involved in the linear approximation of $[y]_i$; consequently, these output bits are called irregular. Neglecting the carry bit and availing the fact that the output bits are composed of independent state bits, the probability $\Pr(Z_\Delta = 1)$ is approximated using Matui's *Piling-up Lemma* [12]. For regular output bits, we obtain

$$\Pr(Z_\Delta = 1) = \frac{1}{2} + 2^3 \cdot \left(\Pr(Y_\Delta = 1) - \frac{1}{2}\right)^4 . \tag{7}$$

Notice that $\epsilon = \Pr(Y_\Delta = 1) - \frac{1}{2}$ is the probability bias. In the case of irregular output bits, one of the four factors ϵ in (7) is substituted by $\epsilon' = \Pr(X_\Delta = 1) - \frac{1}{2}$. Let us consider the case of $\Delta = 3$; it is $\Pr(Z_3 = 1) = 0.5031$ for regular output bits (and a balanced probability for irregular output bits). However, as we neglected the carry bit in this simple model, the above probability is considered as an upper limit. Notice that the carry is also biased and inclines towards absorbing itself. Experiments show that indeed, most of the regular output bits are biased for $\Delta = 3$. We emphasise that higher bits are affected equivalently to lower bits. Due to the integer addition, the exact bias depends on the bit-position. We find the maximum bias for bit-position $i = 1$ with $p' = 0.5003$. A similar result is obtained for $\Delta = 8$ and $i = 0$.

This biased probability is accessible to a cryptanalyst with known plaintext and may be used to distinguish the outcome of the cipher from a uniform random outcome. With the uniform probability $p = 1/2$ and the biased probability $p' = p(1 + q)$, the required data complexity becomes $\mathcal{O}(1/pq^2)$, see Theorem 2 in [11]. Consequently, for $p' = 0.5003$ we expect a data and online time complexity of about 2^{22} (16 MB of keystream); offline time complexity is negligible. We performed a number of experiments (taking all biased bits into account) and verified the predicted complexity, given a small probability of error.

As described above, a variant of this attack even works without taking into account any specific property of the single-cycle S-box. Finally, we mention that the bias of Z_Δ can be transformed in a state-recovery attack by guess-and-determine. In a first step, we guess the least-significant bit-slice $[x^t]_0$, which may be iterated separately. The four corresponding bits are subtracted independently from appropriate output bits in order to construct a modified index variable. Considering (7), we expect the bias to significantly increase for a right guess, and we expect a balanced output for a false guess. After recovering $[x^t]_0$, we may continue with consecutive bit-slices. Considering all available equations, experiments showed that a single bit-slice may be accepted or rejected (with a reasonable probability of error) using 2^{22} iterations. Repeating this for all 2^4 values of a single bit-slice, and for all 2^5 bit-slices, we obtain an overall complexity of about 2^{31}. A similar result has also been obtained by Peyrin and Muller [14].

3.3 Attacking the Stream Cipher TSC-2

In both versions of TSC, the 32 bits of α determine the update of the 128 bits of the state. Hence we may wait for appropriate values of α in order to initiate some attacks. In TSC-2, an interesting case is the minimal-weight parameter $\alpha = 1$, for which only the least significant bit-slice is modified and two similar successive outputs may be detected. The *detector* is an algorithm which takes as input the keystream z and gives out 1 if $\alpha = 1$, and 0 otherwise. The detector can make two types of errors: it can either output 1 when $\alpha \neq 1$ (false positives) or 0 when $\alpha = 1$ (false negatives). The error probabilities are denoted by A and B, respectively.

The complete set of states $\mathcal{U}$ resulting in $\alpha(x^t) = 1$ is given with the conditions $\sum_{i=0}^{3} x_i^t \in \{\texttt{0x00000000}, \texttt{0x80000000}\}$ and $[x^t]_0 \in \{\texttt{0x0}, \texttt{0x3}, \texttt{0x5}, \texttt{0x6}, \texttt{0x9}, \texttt{0xA}, \texttt{0xC}\}$. In the following, let us assume that such a state occurs at time $t = 0$. Hence we have $\alpha^0 = 1$, and only the least significant bit-slice of the state is changed by the mapping $\mathsf{f} : x^0 \to x^1$; consequently, we suppose that the subsequent outputs y^0 and y^1 have low distance. Let us analyse the exemplary integer modular difference $y^0 - y^1$ for $x \in \mathcal{U}$ with $[x^0]_0 = \texttt{0x5}$; we find that $[x^1]_0 = \texttt{0x4}$ and $[x^0]_i = [x^1]_i$ for $i \neq 0$. The output function produces $y^0 = y^1 + 1_{\lll 25} + 1_{\lll 3} + 1_{\lll 12}$ and hence $y^0 - y^1 = \texttt{0x02001008}$. In fact, we find that $y^0 - y^1 = \texttt{const}$ for any $x \in \mathcal{U}$, where the constant $\texttt{const}$ depends only on the least-significant bit-slice $[x^0]_0$ in most of the cases, see Tab. 2. For less than 1% of the states in $\mathcal{U}$, the integer modular difference is not constant because an addition in the output function may cause a carry bit, which propagates from the msb to the lsb due to the cyclic shift. Detection of single constants only would result in a huge amount of false alarms. However, examining Tab. 2, we find a path for the iteration of $[x^0]_0$ with $\texttt{0x6} \to \texttt{0x3} \to \texttt{0xC}$ which is closed in $\mathcal{U}$, meaning that $\alpha^0 = \alpha^1 = \alpha^2 = 1$. Therefore, we may restrict the detector

Table 2. List of output differences for $\alpha = 1$, some of which will be applied in the attack

$[x^0]_0$	$[x^1]_0$	$y^0 - y^1$
0x0	0x5	0xFDBFEFF8
0x3	0xC	0x01C05007
0x5	0x4	0x02001008
0x6	0x3	0xFE3FEFF8
0x9	0x8	0x02001008
0xA	0x1	0xFE002FF9
0xC	0x7	0xFDFFAFF9

to detect only a subset of states $\mathcal{V} \subset \mathcal{U}$, where $\mathcal{V}$ is defined by the conditions $\sum_{i=0}^{3} x_i^t \in \{\texttt{0x00000000}, \texttt{0x80000000}\}$ and $[x^t]_0 \in \{\texttt{0x6}, \texttt{0x3}\}$. The detector takes three successive outputs, computes two differences of consecutive outputs and compares them with the fixed values; if there is a match of both, the detector returns 1, and 0 otherwise. The probability of $x \in \mathcal{V}$ is 2^{-33}, and a false detection due to random outputs[2] occurs with probability 2^{-64}. As the differences are constant almost all the time, the error B (which would increase the running time of the detector) is negligible, too. The time and data complexity is around 2^{33} (no precomputation and negligible memory).

The detector may be transformed in a distinguisher by feeding the detector with a fixed amount of data n. If the detector always returns 0, then the distinguisher returns 0 (random stream); if the detector returns 1 at least once, then the distinguisher returns 1 (keystream produced by TSC-2). The probability of false positives may be neglected, and the probability of false negatives is $B = (1 - 2^{-33})^n$. For $B = 0.05$, we obtain a data complexity of about $n = 2^{34}$.

With a successful detection of $\alpha(x^t) = 1$, we obtain the information $\sum_{i=0}^{3} x_i^t \in \{\texttt{0x00000000}, \texttt{0x80000000}\}$, as well as the value of bit-slice $[x^t]_0$ and the output equation $\mathsf{g}(x^t) = y^t$. This information may be used for a state-recovery attack with a complexity smaller than 2^{96}. However, TSC-2 appears to be seriously injured with our efficient distinguishing attack, and we did not study the state-recovery attack in more detail.

4 Conclusions

In this paper, we examined some specific proposals of stream ciphers based on T-functions. Two proposals by Klimov and Shamir are based on the squaring operation, namely a single word T-function as well as a previously unbroken multi-word T-function with a 256-bit state, both revealing some part of the state. It turned out that the integer differences of consecutive outputs have significant statistical deviation even in the high-order bits. Based on that deviation, we described efficient distinguishing attacks with a 2^{32} data complexity. We conclude that the squaring operation has some undesirable properties when used in the design of T-functions and possibly in other cryptographic primitives. The two proposals by Hong *et al.* have a 128-bit state, which are controlled by a 32-bit parameter and tiny S-boxes. The output function uses some integer additions and rotations. For one of the proposals, we found small loops in the state and in the output produced by the S-box, resulting in a distinguishing attack of complexity 2^{22}. For the other proposal, we wait for an appropriate value of the parameter, which produces some detectable structure in the output. This results in a distinguisher of complexity 2^{34}. We conclude that the small size of the parameter (and potentially also the tiny S-boxes) may be critical, and that the integer additions and rotations in the output functions have a very limited randomizing effect.

[2] In order to increase the set $\mathcal{V}$, we do not make use of the connection of the whole path.

Acknowledgments

This work is supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the Swiss National Science Foundation under grant number 5005-67322. The third author also receives partial funding through Gebert Rüf Stiftung.

We would like to thank Jin Hong and Peter Hellekalek for valuable discussions, and the anonymous reviewers for their helpful comments.

References

1. T. Baignères, P. Junod, and S. Vaudenay. How far can we go beyond linear cryptanalysis ? In P. Lee, editor, *Advances in Cryptology* – ASIACRYPT *2004: 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004. Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. Springer-Verlag, 2004.
2. V. Benony, F. Recher, E. Wegrzynowski, and C. Fontaine. Cryptanalysis of a particular case of Klimov-Shamir pseudo-random generator. In T. Helleseth, D. Sarwate, H.-Y. Song, and K. Yang, editors, *Sequences and Their Applications – SETA 2004, Third International Conference, Seoul, Korea, October 24-28. Revised Selected Papers*, volume 3486 of *Lecture Notes in Computer Science*, pages 313–322. Springer-Verlag, 2004.
3. J. Hong, D. Lee, Y. Yeom, and D. Han. New class of single cycle T-functions and a stream cipher proposal. Presented at *SASC – The State of the Art of Stream Ciphers*, ECRYPT Workshop, October 14-15, Brugge, Belgium, 2004.
4. J. Hong, D. Lee, Y. Yeom, and D. Han. A new class of single cycle T-functions. To appear in H. Gilbert and H. Handschuh, editors, *Fast Software Encryption 2005, 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005. Revised Papers*, volume 3557 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.
5. P. Junod. On the optimality of linear, differential and sequential distinguishers. In E. Biham, editor, *Advances in Cryptology* – EUROCRYPT *2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003. Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, 2003.
6. A. Klimov. *Applications of T-functions in cryptography.* PhD thesis, Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot (Israel), 2004.
7. A. Klimov and A. Shamir. A new class of invertible mappings. In B. Kaliski, Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002: 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002. Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 470–483. Springer-Verlag, 2002.
8. A. Klimov and A. Shamir. Cryptographic applications of T-functions. In *Selected Areas in Cryptography: 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 2003. Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*, pages 248–261. Springer-Verlag, 2004.

9. A. Klimov and A. Shamir. New cryptographic primitives based on multiword T-functions. In B. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2004.
10. A. Klimov and A. Shamir. The TFi family of stream ciphers. Technical note, 2004.
11. I. Mantin and A. Shamir. A practical attack on broadcast RC4. In M. Matsui, editor, *Fast Software Encryption: 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001. Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 152–164. Springer-Verlag, 2002.
12. M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 1993. Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1993.
13. J. Mitra and P. Sarkar. Time-memory trade-off attacks on multiplications and T-functions. In P. Lee, editor, *Advances in Cryptology – ASIACRYPT 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004. Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 468–482. Springer-Verlag, 2004.
14. T. Peyrin and F. Muller. Personal communication, 2005.

A Optimal Distinguishers

In a recent paper, Baignères *et al.* [1] have analysed optimal algorithms (in terms of number of samples) aiming at distinguishing two random sources whose probability distributions are completely known to a cryptanalyst. We briefly recall the framework of Baignères *et al.*

Let D_0 and D_1 be two probability distributions sharing the same support $\mathcal{X}$. We consider the problem of distinguishing these two distributions using ν iid samples. A (possibly computationally unbounded) algorithm $\boldsymbol{\delta}^\nu$ which takes as input a sequence of ν realizations $\boldsymbol{z}^\nu$ distributed according to D where either $\mathsf{D} = \mathsf{D}_0$ or $\mathsf{D} = \mathsf{D}_1$, and outputs 0 or 1 according to its decision, is called a *distinguisher*. It can be fully determined by an acceptance region $\mathcal{A} \subset \mathcal{X}$ such that $\boldsymbol{\delta}^\nu(\boldsymbol{z}^\nu) = 1$ iff $\boldsymbol{z}^\nu \in \mathcal{A}$. The ability to distinguish a distribution from another is usually measured in terms of the *advantage* of the distinguisher and is defined by

$$\mathrm{Adv}_{\boldsymbol{\delta}^\nu} = \left| \Pr_{\mathsf{D}_0^\nu}[\boldsymbol{\delta}^\nu(\boldsymbol{Z}^\nu) = 0] - \Pr_{\mathsf{D}_1^\nu}[\boldsymbol{\delta}^\nu(\boldsymbol{Z}^\nu) = 0] \right| .$$

Hence, the distinguisher can make two types of errors: it can either output 0 when $\mathsf{D} = \mathsf{D}_1$ or 1 when $\mathsf{D} = \mathsf{D}_0$; we will denote these respective error probabilities by α and β, respectively, and the overall error probability is defined as $\pi_{\mathrm{e}} = \frac{1}{2}(\alpha+\beta)$.

In [5] it is shown that it is easy to define explicitly an optimal distinguisher in this precise statistical setting. Indeed, given a fixed overall probability of error, it is sufficient for an optimal distinguisher to count the number $\nu_x(\boldsymbol{z}^n)$ of occurrences of all possible symbols $x \in \mathcal{X}$ in the sample $\boldsymbol{z}^n$, to compute the log-likelihood ratio

$$\mathsf{llr}(\boldsymbol{z}^\nu) = \sum_{x \in \mathcal{X}} \nu_x(\boldsymbol{z}^\nu) \log \frac{\Pr_{\mathsf{D}_0}[x]}{\Pr_{\mathsf{D}_1}[x]} \tag{8}$$

and to output 0 as decision iff $\mathsf{llr}(\boldsymbol{z}^\nu) > 0$. If we assume that the distributions D_0 and D_1 are close to each other, i.e. $\Pr_{\mathsf{D}_0}[x] = \pi_x$ and $\Pr_{\mathsf{D}_1}[x] = \pi_x + \varepsilon_x$ with $|\varepsilon_x| \ll \pi_x$ for all $x \in \mathcal{X}$, then the following result gives a very accurate estimation of the necessary number of samples.

Theorem 1 (Baignères *et al.* [1]). *Let $X_1, \ldots, X_\nu$ be iid random variables defined over $\mathcal{X}$ with probability distribution D, let D_0 and D_1 be two distributions sharing the same support which are close to each other, where $\pi_x = \Pr_{\mathsf{D}_0}[x]$ and $\pi_x + \varepsilon_x = \Pr_{\mathsf{D}_1}[x]$. Let d be a real number defined by*

$$d = \nu \sum_{x \in \mathcal{X}} \frac{\varepsilon_x^2}{\pi_x} \ .$$

Then, the overall probability of error of an optimal distinguisher between D_0 and D_1 is approximately

$$\pi_{\mathrm{e}} \approx \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\sqrt{d}}{2}} e^{-\frac{t^2}{2}} \, dt \ .$$

Baignères *et al.*, based on this result, introduced then what seems to be a natural "measure", named *squared Euclidean imbalance* and denoted $\Delta(\mathsf{D}_0, \mathsf{D}_1)$, between a distribution D_0 and a close distribution D_1 defined by

$$\Delta(\mathsf{D}_0, \mathsf{D}_1) = \sum_{x \in \mathcal{X}} \frac{\varepsilon_x^2}{\pi_x} \ , \tag{9}$$

since $\Delta(\mathsf{D}_0, \mathsf{D}_1)$ is directly linked to the number of samples needed to distinguish both probability distributions with a good success probability.

Introducing a New Variant of Fast Algebraic Attacks and Minimizing Their Successive Data Complexity

Frederik Armknecht[1] and Gwénolé Ars[2]

[1] Theoretische Informatik, Universität Mannheim, 68131 Mannheim, Germany
armknecht@th.informatik.uni-mannheim.de

[2] IRMAR, University of Rennes, Campus de Beaulieu 35042 Rennes, France
gwenole.ars@math.univ-rennes1.fr

Abstract. Algebraic attacks have established themselves as a powerful method for the cryptanalysis of LFSR-based keystream generators (e.g., E_0 used in Bluetooth). The attack is based on solving an overdetermined system of low-degree equations $R_t = 0$, where R_t is an expression in the state of the LFSRs at clock t and one or several successive keystream bits $z_t, \ldots, z_{t+\delta}$.

In fast algebraic attacks, new equations of a lower degree are constructed in a precomputation step. This is done by computing appropriate linear combinations of T successive initial equations $R_t = 0$. The successive data complexity of the attack is the number T of successive equations.

We propose a new variant of fast algebraic attacks where the same approach is employed to eliminate some unknowns, making a divide-and-conquer attack possible. In some cases, our variant is applicable whereas the first one is not.

Both variants can have a high successive data complexity (e.g., $T \geq 8.822.188$ for E_0). We describe how to keep it to a minimum and introduce suitable efficient algorithms for the precomputation step.

Keywords: fast algebraic attacks, stream ciphers, linear feedback shift registers, Bluetooth

1 Introduction

Keystream generators are designed for online encryption of secret plaintext bitstreams M passing an insecure channel. Depending on a secret key K, they produce a regularly clocked bitstream called the keystream $Z = (z_1, z_2, \ldots)$, $z_i \in \mathbb{F}_2$. M is encrypted by XORing both streams termwise. A legal receiver decrypts by applying the same procedure.

Many keystream generators consist of combining several linear feedback shift registers (LFSRs) and possibly some additional memory. One example is the E_0 keystream generator which is part of the Bluetooth standard [6]. An LFSR is a finite automaton which produces a bitstream of arbitrary length depending on its initial state. LFSRs are very efficient in hardware and can be designed such that the produced bitstream has maximum period and good statistical properties. Many different approaches to the cryptanalysis of LFSR-based stream ciphers were discussed in literature (e.g., time-memory-tradeoff [5], fast correlation attacks [17] or BDD-based attacks [13]). For some keystream generators, algebraic attacks outmatched all previously known attacks [8,2,7]. They consist of finding and solving a system of (low-degree) equations in the key bits and the known keystream bits. More precisely, the equations are of the form $R_t = 0$ where R_t is an expression in the state of the LFSRs at clock t and one or several successive keystream bits $z_t, \ldots, z_{t+\delta}$.

If the system is overdetermined, it can be solved by linearization: each occurring monomial is replaced by a new variable, giving a system of linear equations. If all equations are of degree $\leq d$ and $n := |K|$ denotes the key size, then the number of monomials is $\approx \binom{n}{d}$. Thus, the time effort is $\approx \binom{n}{d}^3$ (using Gaussian elimination), the amount of space is $\approx \binom{n}{d}^2$ and the data complexity is $\geq \binom{n}{d}$.

The idea of fast algebraic attack is to find linear combinations $\bigoplus_{i=0}^{T} \lambda_i R_{t+i}$ of the initial equations to obtain a new equation of a lower degree $e < d$ in a precomputation step. As this reduces the number of monomials, the time for computing the solution drops. In this paper, we focus on this precomputation step. We propose a new variant of this approach. The difference is that the number of unknowns in reduced instead of the degree. We present an example where our attack is faster than all other algebraic attacks proposed so far.

Both variants work only if the attacker knows the value of the keystream bits involved in $R_t, \ldots, R_{t+T}$. We introduce the term *successive data complexity* for $T = T(R)$. We present a theory which allows to specify the exact minimum successive data complexity for both attacks. The amount of data is only slightly decreased indeed, but it might help to make fast algebraic attacks more practically. In particular, we give efficient algorithms to achieve the miminum data complexity in precomputation step.

The paper is organized as follows: In Section 2, we provide some definitions and basic results needed for the rest of the paper, and we describe fast algebraic attacks in Section 3. We introduce a variant of these attacks in Section 4. In Section 5, theory and methods are developed for efficient precomputation steps having the minimum successive data complexity. Finally, we give a short conclusion in Section 6.

2 Definitions and Basics Results

In this Section, we provide some definitions and facts, used in the paper.

For an integer $\alpha = \sum_i \alpha_i \cdot 2^i$, $\alpha_i \in \{0, 1\}$, we define its weight $wt(\alpha) := \sum_i \alpha_i$. For a vector $(\alpha^{(1)}, \ldots, \alpha^{(m)})$ of integers, we extend this definition to $wt(\alpha^{(1)}, \ldots, \alpha^{(n)}) = \sum wt(\alpha^{(i)})$.

For positive integers $n_1, \ldots, n_m$, let $X_i := (x_{i,1}, \ldots, x_{i,n_m})$. For $\alpha_i = \sum_{j=0}^{n_i-1} \alpha_{i,j} 2^j$, we define $X_i^{\alpha_i} := \prod_{j=1}^{n_i} x_{i,j}^{\alpha_{i,j}}$ and for $\alpha = (\alpha_1, \ldots, \alpha_m)$ the expression

$X^\alpha := \prod_{j=1}^m X_j^{\alpha_j}$. Each Boolean function $F(X_1, \ldots, X_m)$ in $n := n_1 + \ldots + n_m$ unknowns has a unique algebraic expression $\bigoplus_{\alpha=(\alpha_1,\ldots,\alpha_m)} c_\alpha \cdot X^\alpha$ with $c_\alpha \in \{0,1\}$. The uniqueness allows to define two different types of degree:

$$\deg(F) := \max\{wt(\alpha_1, \ldots, \alpha_m) \mid c_{(\alpha_1,\ldots,\alpha_m)} \neq 0\}$$
$$\deg_{X_i}(F) := \max\{wt(\alpha_i) \mid c_{(\alpha_1,\ldots,\alpha_m)} \neq 0\}$$

If $deg_{X_i}(F) = 0$, then F is independent of the variables in X_i.

For $\omega \in \mathbb{F}_{2^n}$, we define its minimal polynomials m_ω to be the unique non-zero polynomial $m(x) \in \mathbb{F}_2[x]$ of the lowest degree such that $m(\omega) = 0$. For a set $\Omega \subseteq \mathbb{F}_{2^n}$ let m_Ω denote the non-zero polynomial of the lowest degree such that $m_\Omega(\omega) = 0$ for all $\omega \in \Omega$. The following facts are well known:

Lemma 1. *Let $\Omega_1, \Omega_2 \subseteq \mathbb{F}_{2^n}$. Then $m_{\Omega_1 \cap \Omega_2} = gcd(m_{\omega_1}, m_{\omega_2})$. If m_{Ω_1} and m_{Ω_2} are co-prime, then $m_{\Omega_1 \cup \Omega_2} = m_{\Omega_1} \cdot m_{\Omega_2}$.*

Theorem 1. *Let $f(x) \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n and $\omega, \omega' \in \mathbb{F}_{2^n}$ be two roots. Then $\omega = (\omega')^{2^k}$ for an appropriate k.*

The proofs for the following claims can be found in [14].

A sequence $Z = (z_t)$ over $\mathbb{F}_2$ is called a linear recurring sequence if coefficients $\lambda_0, \ldots, \lambda_{T-1} \in \{0,1\}$ (not all zero) exist such that $\bigoplus \lambda_i z_{t+i} = 0$ is true for all values $t \geq 1$. In this case, $\bigoplus \lambda_i x^i \in \mathbb{F}_2[x]$ is called a characteristic polynomial of the sequence Z. Amongst all characteristic polynomials of Z there exists one unique polynomial m_Z which has the lowest degree. We will call it the minimal polynomial of Z. A polynomial $f(x) \in \mathbb{F}_2[x]$ is a characteristic polynomial of Z if and only if m_Z divides $f(x)$.

Let $p(x) = \sum_{i=0}^n \lambda_i \cdot x^i \in \mathbb{F}_2[x]$. The companion matrix L_p of $p(x)$ is defined by

$$L_p := \begin{pmatrix} 0\,0 \ldots 0 & a_0 \\ 1\,0 \ldots 0 & a_1 \\ \vdots\,\vdots\,\ddots\,\vdots & \vdots \\ 0\,0 \ldots 1 & a_{n-1} \end{pmatrix}.$$

A linear feedback shift register (LFSR) is a regularly clocked finite state machine together with a polynomial $p(x) \in \mathbb{F}_2[x]$ of degree n and an internal state $S \in \{0,1\}^n$. Let $L := (L_p)^t$ be the transpose of the companion matrix. At each clock, it outputs the first Bit of S and updates S to $L \cdot S$. The produced keystream is a linear recurring sequence with minimal polynomial $p(x)$. If $p(x)$ is primitive, the sequence has the maximum period $2^n - 1$. p is also called the feedback polynomial of the LFSR and L the corresponding feedback matrix.

3 Fast Algebraic Attacks

In this Section, we describe (fast) algebraic attacks on LFSR-based keystream generators. An LFSR-based keystream generator consists of m LFSRs of lengths

$n_1, \ldots, n_m$ and $\ell \geq 0$ additional memory bits. If $\ell = 0$, we speak of simple combiners, otherwise of combiners with memory. We denote by $K = (K_1, \ldots, K_m)$ with $K_i \in \{0,1\}^{n_i}$ the initial states of the LFSRs and by $M_t \in \{0,1\}^{\ell}$ the content of the memory bits at clock t. Let L_i be the feedback matrix of the i-th LFSR and $L := diag(L_1, \ldots, L_m)$. At each clock t, an output bit z_t is produced by $z_t = f(L^t \cdot K, M_t)$ and the memory is updated to M_{t+1} depending on $L^t \cdot K$ and M_t. It is assumed that an attacker knows everything except K and M_t, $t \geq 0$. An algebraic attack works as follows: First find a Boolean function $R \neq 0$ such that

$$R_t := R(L^t \cdot K, z_t, \ldots, z_{t+\delta}) = 0 \tag{1}$$

is true for all clocks t. We will call (1) a valid equation if it is true for all K, t and corresponding keystream bits $z_t, \ldots, z_{t+\delta}$. The case of simple combiners has been examined in [8,16] and the case of combiners with memory in [2]. A unified treatment of both cases is given in [4]. The adversary can use (1) to set up a system of equations, describing K subject to of the observed keystream bits z_t. The final step is to recover K by solving this system of equations.

Although this task is difficult in general, in this special situation an attacker can exploit that $\deg_K(R_t) \leq \deg_K(R_0) := d$ for all t. Therefore, the number μ of different monomials with unknowns coming from K is upper bounded by $\sum_{i=0}^{d} \binom{n}{i} \approx \binom{n}{d}$. If enough keystream bits are known, the number of linearly independent equations (1) equals μ. Substituting each monomial by a new variable results in a system of linear equations in μ unknowns which can be solved by Gaussian elimination or more refined methods like Strassen's one [20]. The computational complexity is in $O\left(\binom{n}{d}^{\epsilon}\right)$ with $\epsilon \leq 3$, which is polynomial in the key size n but exponential in the degree d. The amount of space is in $O\left(\binom{n}{d}^{2}\right)$.

Two different strategies to accelerate this attack are straightforward. The first one would be to find a faster method to solve this special kind of system of equations (e.g., using Gröbner bases [10]). We will not pursue this approach in this paper. Instead, we will focus on the second strategy, i.e. to reduce the number μ of monomials. This idea has been used in fast algebraic attacks (FAA), introduced in [9]. In a precomputation step, appropriate linear combinations of the initial equations $R_t = 0$ are computed to get new ones of lower degree $e < d$.

FAA do not work in general. They require that (1) can be rewritten as

$$0 = R(L^t \cdot K, z_t, \ldots, z_{t+\delta}) = F(L^t \cdot K) \oplus G(L^t \cdot K, z_t, \ldots, z_{t+\delta}) \tag{2}$$

where $e := \deg_K(G) < \deg_K(R) = d$. For example, this condition is true for the three ciphers E_0, Toyocrypt and LILI-128 [9]. Then, the attacker needs to find coefficients $\lambda_0, \ldots, \lambda_{T-1} \in \{0,1\}$ such that

$$\bigoplus_{i=0}^{T} \lambda_i \cdot F(L^{t+i} \cdot K) = 0 \quad \forall t, K. \tag{3}$$

In general, the coefficients λ_i depend on K. But if the feedback polynomials of the m LFSRs are co-prime[1], the λ_i are the same for all K. Actually, this is even

[1] This is the case for most keystream generators.

true under weaker conditions (see [3]). Hence, for the rest of the paper we will assume the λ_i's to be independent of K.

Using (2) and (3), the equation

$$0 = \bigoplus_{i=0}^{T} \lambda_i R_{t+t} = \bigoplus_{i=0}^{T} \lambda_i \cdot G(L^{t+i} \cdot K, z_{t+i}, \ldots, z_{t+i+\delta})$$

is valid and of degree $e < d$. By repeating this procedure for several clocks t, the attacker can transform the system of equations of degree d given by (1) into a new one of degree $e < d$:

$$\underbrace{\begin{array}{rcl} 0 &=& R(K, z_0, \ldots, z_\delta) \\ 0 &=& R(L \cdot K, z_1, \ldots, z_{\delta+1}) \\ & \ldots & \end{array}}_{\text{degree } d} \mapsto \underbrace{\begin{array}{rcl} 0 &=& \bigoplus_{i=0}^{T} \lambda_i G(L^i \cdot K, z_i, \ldots, z_{i+\delta}) \\ 0 &=& \bigoplus_{i=0}^{T} \lambda_i G(L^{1+i} \cdot K, z_{1+i}, \ldots, z_{1+i+\delta}) \\ & \ldots & \end{array}}_{\text{degree } e} \quad (4)$$

Note that this decreases the degree and hence the number of monomials, speeding up the solving step enormously. Also the space requirements are reduced from roughly $\binom{n}{d}^2$ to $\binom{n}{e}^2$. On the other hand, the knowledge of the keystream bits from T successive equations is required to construct one equation of degree e. We will call this term the *successive data complexity*. Observe that it depends on the chosen function R in (1). For example, in [3] it was estimated, based on tests with smaller LFSRs, that $T = 8.822.188$ for the Bluetooth keystream generator E_0, to reduce the degree from $d = 4$ to $e = 3$.

One advantage of FAA is that the precomputation needs to be done only once. The coefficients λ_i can be computed with the Berlekamp-Massey algorithm [15]. In certain cases, faster and parallelizable methods exist [3]. Somewhat surprising, the direct insertion of the observed keystream bits into the system of equations can dominate the attack complexity but this problem has been solved in [12].

In the next section, we introduce a second method similar to FAA, which allows a divide-and-conquer approach. Instead of reducing the degree, the new equations are independent of one or several LFSRs. Hence, it is possible to first reconstruct the remaining LFSRs and afterwards the ones which have been canceled out. Cases exist where our attack works whereas other methods fail to reduce the complexity. As same as manipulating equations (2) for FAA, we need equations with a special structure. Applying similar methods form those in [4,16] give us such equations, but it is not the subject of this paper.

4 A New Variant of Fast Algebraic Attacks

As explained in Section 3, fast algebraic attacks are based on processing the system of equations in a precomputation step in order to decrease the degree of the equations. In this section, we show that a similar approach may be used to reduce the number of unknowns. Before we give a general description, we motivate the attack by an example.

The example uses five LFSRs of lengths $n_1, \ldots, n_5$ filtered by a memory function. Let $x_t^{(i)}$ denote the output of the ith LFSR at clock t and $n := n_1 + \ldots + n_5$ the key size. The output z_t for $t \geq 0$ is computed as follows:

$$\begin{aligned} z_t &:= x_t^{(1)} \oplus x_{t-1}^{(1)} \cdot (\sigma_{t-1}^2 \oplus \sigma_t^2) \oplus \sigma_t^3 \oplus \sigma_{t-1}^2 \oplus C_t \cdot \sigma_{t-1}^3 \\ C_{t+1} &:= z_t \end{aligned}$$

where C_0 is some initial memory of the function and $\sigma_t^d := \bigoplus_{2 \leq i_1 < \ldots < i_d \leq 5} x_t^{(i_1)} \cdot \ldots \cdot x_t^{(i_d)}$ is the d-th elementary symmetric polynomial in the outputs $x_t^{(2)}, x_t^{3)}, x_t^{(4)}, x_t^{(5)}$. We have an algebraic relation independent of the function memory, for $t \geq 1$:

$$z_t := x_t^{(1)} \oplus x_{t-1}^{(1)} \cdot (\sigma_{t-1}^2 \oplus \sigma_t^2) \oplus \sigma_t^3 \oplus \sigma_{t-1}^2 \oplus z_{t-1} \cdot \sigma_{t-1}^3 \tag{5}$$

Equation (5) is of degree 3 in the key bits, which yields an algebraic attack with complexity $O\left(\binom{n}{3}^\epsilon\right)$. We have tested with the usual methods[2] that no quadratic relations exist over two clocks. Because of the term $z_{t-1} \cdot \sigma_{t-1}^3$, it is not possible to split (5) as shown in (2). Therefore, a FAA as described in [9] is not applicable in this case.

Furtheron, we checked that what we would call "local divide-and-conquer" attack is possible (e.g., see [11]): equations exist which are independent of $x_t^{(i)}$ and $x_{t-1}^{(i)}$ for one $i \in \{2, 3, 4, 5\}$, but there is no equation independent of variables $x_t^{(1)}$ and $x_{t-1}^{(1)}$. But as the lowest degree of these equations is 4, this approach would increase the complexity instead of lessen it. Altogether, it seems that the naive algebraic attack is the best algebraic attack in this case.

Actually, one can do better. Similar to (2), we rewrite (5) to

$$0 = \underbrace{x_t^{(1)} \oplus x_{t-1}^{(1)} \cdot (\sigma_{t-1}^2 \oplus \sigma_t^2)}_{=:F_t} \oplus \underbrace{z_t \oplus \sigma_t^3 \oplus \sigma_{t-1}^2 \oplus z_{t-1} \cdot \sigma_{t-1}^3}_{=:G_t}$$

G_t is independent of the outputs of LFSR 1. As $F_0, F_1, \ldots$ is a linear recurring sequence, coefficients $\lambda_0, \ldots, \lambda_T$ exist with $\oplus_{i=0}^T \lambda_i F_{t+i} = 0$ for all K and t. Then, $0 = \bigoplus_{i=0}^T \lambda_i R_{t+i} = \bigoplus_{i=0}^T \lambda_i G_{t+i}$ is a valid equation of degree 3 which is independent of LFSR 1. This reduces the number of monomials from $O\left(\binom{n}{3}\right)$ to $O\left(\binom{n-n_1}{3}\right)$ and the computational complexity from $O\left(\binom{n}{3}^\epsilon\right)$ to $O\left(\binom{n-n_1}{3}^\epsilon\right)$. If the feedback polynomials are pairwise co-prime, then the successive data complexity is $T = n_1 + n_1 . \sum_{2 \leq i < j \leq 5} n_i n_j$ (see [14]). For $(n_1, \ldots, n_5) = (33, 27, 26, 25, 17)$ and $\epsilon = 3$, the time and space efforts for different algebraic attacks are displayed in Table 1.

We give now a more general description of this approach. Required is a valid equation (1) which can be split as shown here:

$$0 = F(L^t \cdot K) \oplus G(\hat{L}_j^t \cdot \hat{K}_j, z_t, \ldots, z_{t+\delta}). \tag{6}$$

[2] A description can be found in [4].

Table 1. Different algebraic attacks against the given example

Attack	degree	#unknowns	time	memory
algebraic attack	3	128	$\approx 2^{55}$	$\approx 2^{37}$
fast algebraic attack	/	/	/	/
local divide-and-conquer	4	128-33=95	$\approx 2^{65}$	$\approx 2^{43}$
our attack	3	128-33=95	$\approx 2^{52}$	$\approx 2^{35}$

$\hat{L}_j$ resp. $\hat{K}_j$ denote the companion matrix resp. the key where the part belonging to the jth LFSR is left out. More precisely, it is $\hat{L}_j := diag(L_1, \ldots, L_{j-1}, L_{j+1}, \ldots, L_m)$ and $\hat{K}_j := (K_1, \ldots, K_{j-1}, K_{j+1}, \ldots, K_m)$. The next step is to compute coefficients λ_i such that $\bigoplus_{i=0}^{T} \lambda_i F(L^{t+i} \cdot K) = 0$ for all t and K. As for fast algebraic attacks, the Berlekamp-Massey algorithm can be used, but more refined methods will be presented in Section 5.4. Observe that $\bigoplus_{i=0}^{T} \lambda_i G(\hat{L}_j^{t+i} \cdot \hat{K}_j, z_{t+i}, \ldots, z_{t+i+\delta}) = 0$ is a valid equation and independent of K_j. Repeating this step for several t gives a system of equations which is independent of K_j:

$$\underbrace{\begin{array}{rcl} 0 &=& R(K, z_0, \ldots, z_\delta) \\ 0 &=& R(L \cdot K, z_1, \ldots, z_{\delta+1}) \\ \ldots \end{array}}_{n\ unknowns} \mapsto \underbrace{\begin{array}{rcl} 0 &=& \bigoplus_i \lambda_i G(\hat{L}_j^i \cdot \hat{K}_j, \ldots) \\ 0 &=& \bigoplus_i \lambda_i G(\hat{L}_j^{1+i} \cdot \hat{K}_j, \ldots) \\ \ldots \end{array}}_{n-n_j\ unknowns} \qquad (7)$$

This reduces the number of computation steps from (roughly) $\binom{n}{d}^\epsilon$ to $\binom{n-n_j}{d}^\epsilon$ and the amount of space from $\binom{n}{d}^2$ to $\binom{n-n_j}{d}^2$. Afterwards, the values of K_j can be easily reconstructed or even be guessed.

5 Minimizing the Successive Data Complexity

5.1 Preliminary Notes

Both attacks described in sections 3 and 4 are based on replacing given equations $R_t = 0$ by appropriate linear combinations $\bigoplus_i \lambda_i R_{t+i}$. Appropriate means that $\bigoplus_i \lambda_i F_{t+i}$ has some desired property P where $R_t = F_t \oplus G_t$. To perform the precomputation step, it is necessary to have a (preferably efficient) method to find such coefficients. We propose the following treatment of the problem:

1. Use Φ from Section 5.2 to compute $\Phi(F) = H \in \mathbb{F}[Y_1, \ldots, Y_m]$ with $\mathbb{F}$ an extension field of $\mathbb{F}_2$.
2. Specify a set $\Omega = \mathbb{F}$ depending on H and the desired property P as described in Section 5.3.
3. Determine a polynomial $p(x) = \bigoplus_i \lambda_i x^i$ such that $p(\omega) = 0$ for all $\omega \in \Omega$.

In Section 5.3, we will show that for all vectors $(\lambda_0, \ldots, \lambda_T) \in \{0,1\}^{T+1}$, it is

$$\bigoplus_i \lambda_i F_{t+i} \text{ has property } P \iff p(x) = \bigoplus_i \lambda_i x^i \ : \ p(\omega) = 0 \ \forall \omega \in \Omega.$$

The degree T of $p(x)$ determines the number of successive equations. If it is possible to compute the minimal polynomial of Ω efficiently, then this gives immediately the coefficients for the minimum successive data complexity.

In Section 5.4, we will describe efficient algorithms for computing these polynomials. We will see that these methods do not require the computation of H or Ω. Still, we need to specify them once to to prove the correctness of the algorithms.

5.2 Definition of the Function Φ

In this Section, we develop a theory which will make it possible to specify the minimum successive data complexity and to derive efficient methods for computing the associated coefficients. The strategy is to describe $F(L^t \cdot K)$ as a multivariate polynomial in an extension field of $\mathbb{F}_2$. The theory presented in this Section is similar to that discussed in [3] but develops it further. The proofs can be found in Appendix A.

The first step is to introduce a special bijection $\mathbb{F}_2^n \to \mathbb{F}_{2^n}$ and extend it to $\mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m} \to \mathbb{F}_{2^{n_1}} \times \ldots \times \mathbb{F}_{2^{n_m}}$. Although we describe the results in their generality, we should keep in mind that the situations are motivated by Sections 3 and 4. The polynomials p correspond to the feedback polynomials of the LFSRs and L to their feedback matrices.

Theorem 2. *Let $p(x) = \bigoplus_{i=0}^{n} a_i \cdot x^i \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n and with root ω. Furtheron, let $L \in GL_n(\mathbb{F}_2)$ be the transpose of its companion matrix and $X := (x_0, \ldots, x_{n-1})$. There exists a $\mathbb{F}_2$-linear bijection $\varphi_L : \mathbb{F}_2^n \to \mathbb{F}_{2^n}$ with $\varphi_L(L^i \cdot X) = \omega^i \varphi_L(X)$.*

Corollary 1. *Let $p_1(x), \ldots, p_m(x) \in \mathbb{F}_2[x]$ be irreducible polynomials of degree $n_1, \ldots, n_m$ and with roots $\omega_1, \ldots, \omega_m$ respectively. Let $L_1, \ldots, L_m$ be the transposed of their companion matrices. Then there exists a linear bijection $\varphi := \varphi_{L_1,\ldots,L_m} : \mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m} \to \mathbb{F}_{2^{n_1}} \times \ldots \times \mathbb{F}_{2^{n_m}}$ such that for all $\mathcal{X} := (X_1, \ldots, X_m) \in \mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m}$ and $(\tilde{X}_1, \ldots, \tilde{X}_m) = \varphi(\mathcal{X})$, we have*

$$\varphi(L_1^i \cdot X_1, \ldots, L_m^i \cdot X_m) := (\varphi_{L_1}(X_1), \ldots, \varphi_{L_m}(X_m)) = (\omega_1^i \tilde{X}_1, \ldots, \omega_m^i \tilde{X}_m).$$

The function φ allows to give an alternative description of $\deg(F)$ and $\deg_{K_j}(F)$:

Theorem 3. *Let $p_j(x) \in \mathbb{F}_2[x]$, $1 \le j \le m$, be irreducible polynomials of degrees n_j and $\mathbb{F}_{2^{n'}}$ be their splitting field. There exists a linear injection $\Phi = \Phi_{p_1,\ldots,p_m}$:*

$$\Phi : \{F \in \mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m} \to \mathbb{F}_2\} \hookrightarrow \{H \in \mathbb{F}_{2^{n'}}[Y_1, \ldots, Y_m]/\langle Y_i^{2^{n'}} - Y_i, \forall i\rangle | H^2 = H\}$$
$$F \mapsto \sum_{0 \le \alpha_1,\ldots,\alpha_m \le 2^{n'}-1} c_{\alpha_1,\ldots,\alpha_m} \cdot Y_1^{\alpha_1} \cdot \ldots \cdot Y_m^{\alpha_m}$$

such that

$$\deg(F) = \max\{wt(\alpha_1, \ldots, \alpha_m) \,|\, c_{\alpha_1,\ldots,\alpha_m} \neq 0\} \text{ and}$$
$$\deg_{X_j}(F) = \max\{wt(\alpha_j) \,|\, c_{\alpha_1,\ldots,\alpha_m} \neq 0\}.$$

5.3 Specifying the Minimum Successive Data Complexity

The attacks described in Sections 3 and 4 are both based on the following: given a Boolean function F and a feedback matrix L, find coefficients $\lambda_0, \ldots, \lambda_T$ such that $\bigoplus \lambda_j F(L^{t+j} \cdot K)$ has a certain property. We show that this is equivalent to finding certain Boolean functions $p(x) = \bigoplus \lambda_j x^i \in \mathbb{F}_2[x]$. Because of $T = \deg(p)$, the polynomial having the lowest possible degree determines the minimum successive data complexity. Table 2 gives an overview of the polynomials considered in this Section.

The polynomials m_F and m_d belong to the fast algebraic attacks as introduced in [9]. If F contains products of the outputs coming from one LFSR, then some of the F_t have monomials of degree $< d$. If the degree is $\leq e$, then there is no need to cancel them out. It suffices to eliminate all monomials of degree between $e+1$ and d. Therefore, we introduce the polynomials $m_{(e,d],F}$ resp. $m_{(e,d],d}$. Because of $\deg(m_{(e,d],F}) \leq \deg(m_F)$ and $\deg(m_{(e,d],d}) \leq \deg(m_d)$, this can reduce the successive data complexity. A toy example illustrating this effect can be found in Appendix B. In both cases, the complexity of solving the system of equations is asymptotically the same.[3] The polynomials $m_{X_i,F}$ and $m_{X_i,d}$ give the coefficients for the divide-and-conquer attack described in Section 4.

Table 2. Description of the minimal polynomials discussed in Section 5.3

Polynomial $\bigoplus_{j=0}^{T} \lambda_j x^j$	Property of $\bigoplus_{j=0}^{T} \lambda_j F(L^{t+j} \cdot K)$
m_F	equal to zero
m_d	equal to zero for any F with $\deg(F) \leq d$
$m_{(e,d],F}$	no monomials of degree $\in (e,d]$
$m_{(e,d],d}$	no monomials of degree $\in (e,d]$ for any F with $\deg(F) \leq d$
$m_{X_i,F}$	independent of K_i
$m_{X_i,d}$	independent of K_i for any F with $\deg(F) \leq d$

Theorem 4. *Let $\Phi(F) = H = \sum_\alpha c_\alpha Y^\alpha$ with $Y^\alpha := Y_1^{\alpha_1} \cdot \ldots \cdot Y^{\alpha_m}$ and $\alpha = (\alpha_1, \ldots, \alpha_m)$. Because of $\Phi(F(L^i \cdot (X_1, \ldots, X_m)) = H(\omega_1^i Y_1, \ldots, \omega_m^i Y_m)$, $\Phi(F(L^i \cdot (X_1, \ldots, X_m)) = \sum_\alpha c_\alpha \omega^{i \cdot \alpha} Y^\alpha$. We define*

$$\Omega_F := \{\omega^\alpha | c_\alpha \neq 0\},\ \Omega_{(e,d],F} := \{\omega^\alpha | c_\alpha \neq 0,\ e < wt(\alpha) \leq \deg(F)\}$$

Then for $p(x) = \bigoplus_{i=0}^{T'} \lambda_j x^j$, we have equivalences:

$$\begin{aligned} \bigoplus \lambda_j F(L^{t+j} \cdot K) = 0 &\iff p(\omega') = 0 \quad \forall \omega' \in \Omega_F \\ \deg_K(\bigoplus \lambda_j F(L^{t+j} \cdot K)) \leq e &\iff p(\omega') = 0 \quad \forall \omega' \in \Omega_{(e,d],F} \end{aligned} \tag{8}$$

[3] In the concrete case, it may happen that more different monomials occur in the second approach. This would result in a slightly higher complexity.

In particular, for the unique minimal polynomials m_F resp. $m_{(e,d],F}$ of the sets Ω_F resp. $\Omega_{(e,d],F}$, the values $T' = \deg(m_F)$ resp. $T = \deg(m_{(e,d],F})$ are the minimum successive data complexity. Furtheron, this means $T \leq T'$.

The Theorem shows that the minimum successive data complexity in a fast algebraic attack is $T = \deg(m_{(e,d],F})$. If F consists of products of outputs coming from the same LFSR, $m_{(e,d],F}$ is different to m_F (see example given in Appendix B). The precomputation steps proposed so far compute m_F, which in certain cases results in a higher successive data complexity than necessary. A general estimation for T' and T are $\binom{n}{0} + \ldots + \binom{n}{d}$ and $\binom{n}{e+1} + \ldots + \binom{n}{d}$, respectively. To solve completely the new equations with linear algebra and find the key bits, we need $\binom{n}{0} + \ldots + \binom{n}{e}$ equations. Then the total data complexity can be bounded between $T + \binom{n}{0} + \ldots + \binom{n}{e}$ and $T\left(\binom{n}{0} + \ldots + \binom{n}{e}\right)$. The lower bound implies that we have only successive data bits, it correspond to the minimum bits needed to achieve the algebraic attack and the second that the equations come from independant output bits.

For example, in the case of LILI-128 [19], a conventional fast algebraic attack would require the knowledge of $T \approx 2,559,195$ successive keystream bits for one degree-3-equation. Our approach shows that the same is possible with $T' - T \approx 117,569$ fewer keystream bits. A more concrete example is the Bluetooth keystream generator E_0. In [3], it was stated that $T \leq 8,822,188$ successive keystream bits are necessary for one degree-3-equation. We computed that this is also possible for at least 326,080 fewer bits (see Appendix C).

Inspired by the example of E_0 where $e = 3$ and $d = 4$, we have implemented some simulations to compare T with T' and the number of non-zero coefficients λ'_j and λ_j. Observe that the last values reflect how many equations have to be summed up. The examples have been done with only one LFSR defined by L and a fixed random F which is the sum of monomials of degree 4. The results can be found in Table 3.

The results demonstrate that if n is not too small, T' and T are comparatively close. Hence, for the asymptotic complexity of the attack, using the minimum possible T has a rather minor influence. On the other hand, we will show in Section 5.4 that $m_{(e,d],F}$ can be computed efficiently. Therefore, we see no reason why an attacker should use more successive keystream bits than necessary.

Table 3. Comparison of Berlekamp-Massey (upper row) with our method (lower row)

Method	n	8	9	10	11	12	13	14	15	16
Berlekamp-	succ. data complexity	161,2	255	384,8	561	793	1092	1469,2	1940	2516
Massey	$\#\{\lambda'_j \neq 0\}$	84,3	132,6	195,2	278,4	413,6	543	741,7	963,4	1253,5
Our	succ. data complexity	71	126	210	330	495	715	1001	1365	1820
Method	$\#\{\lambda_j \neq 0\}$	31	61,3	102,1	159,1	243	351,7	502,3	677,6	917

In some cases, it may be difficult to calculate the sets Ω_F resp. $\Omega_{(e,d],F}$, or F might be an unknown function of degree d. The following Corollary specifies "general" polynomials $\bigoplus \lambda'_j x^j$ and $\bigoplus \lambda_j x^j$ such that equations (8) hold for any Boolean function F of degree $\le d$:[4]

Corollary 2. *Let $p_1(x), \ldots, p_m(x) \in \mathbb{F}_2[x]$ be irreducible polynomials of degree $n_1, \ldots, n_m$, L the block matrix of the transposed companion matrices of all p_i and $\omega = (\omega_1, \ldots, \omega_m)$ where ω_i is a root of p_i. Furtheron, let $\Omega_d := \{\omega^\alpha | 0 < wt(\alpha) \le d\}$ resp. $\Omega_{(e,d],d} := \{\omega^\alpha | e < wt(\alpha) \le d\}$. Let $p(X) := \bigoplus_{j=0}^T \lambda_j x^j$. Then, for any Boolean function F of degree $\le d$, we have equivalences:*

$$\begin{aligned} \textstyle\bigoplus_j \lambda_j F(L^{t+j} \cdot K) = 0 &\iff p(\omega') = 0 \quad \forall \omega' \in \Omega_d \\ \deg(\textstyle\bigoplus_j \lambda_j F(L^{t+j} \cdot K) \le e &\iff p(\omega') = 0 \quad \forall \omega' \in \Omega_{(e,d],d} \end{aligned}$$

Therefore, the minimal polynomials m_d and $m_{(e,d],F}$ of the sets Ω_d and $\Omega_{(e,d],F}$ specify the minimum successive data complexity and the coefficients λ_i.

So far, we proved that optimal coefficients λ_i for reducing the degree can be derived by computing certain minimal polynomials. The following Theorem shows that the same is possible for the divide-and-conquer attack described in Section 4.

Theorem 5. *Let $F \in \mathbb{F}_2[X_1, \ldots, X_m]$, $n := \sum_{j=1}^m |X_i$, $L \in GL_n(\mathbb{F}_2)$ and $\Phi(F) =: H = \sum_\alpha c_\alpha Y^\alpha$. We set $\Omega_{X_i,F} := \{\omega^\alpha | c_\alpha \neq 0, wt(\alpha) \le \deg(F), \alpha_i \neq 0\}$. Then for $p(X) := \bigoplus_{j=0}^T \lambda_j x^j$, we have the equivalence:*

$$\deg_{X_i}\left(\bigoplus_{j=0}^{T} \lambda_j F(L^{t+j} \cdot K)\right) = 0 \iff p(\omega') = 0 \quad \forall \omega' \in \Omega_{X_i,F}.$$

The left side means that $\bigoplus_{j=0}^T \lambda_j F(L^{t+j} \cdot K) = 0$ is a valid equation independent of X_i.

Let $p_1(x), \ldots, p_m(x) \in \mathbb{F}_2[x]$ be irreducible polynomials of degree $n_1, \ldots, n_m$, L the block matrix of the transposed companion matrices of all p_i and $\omega = (\omega_1, \ldots, \omega_m)$ where ω_i is a root of p_i. Furtheron, let $\Omega_{X_i,d} := \{\omega^\alpha | wt(\alpha) \le d, \alpha_i \neq 0\}$. Then for any Boolean function $F \in \mathbb{F}_2[X_1, \ldots, X_m]$ of degree $\le d$, we have the equivalence:

$$\deg_{X_i}\left(\bigoplus_{j=0}^{T} \lambda_j F(L^{t+j} \cdot K)\right) = 0 \iff p(\omega') = 0 \quad \forall \omega' \in \Omega_{X_i,d}.$$

Again do the minimal polynomials $m_{X_i,F}$ resp. $m_{(e,d],F}$ of the sets $\Omega_{X_i,F}$ resp. $\Omega_{X_i,d}$ specify the minimum successive data complexity and the coefficients λ_i.

[4] Of course, it does not longer guarantee the minimality of T' and T, respectively.

5.4 Efficient Precomputation Steps with the Minimum Successive Data Complexity

In the previous Section, we showed that finding the appropriate linear combination for the precomputation steps is equivalent to computing certain minimal polynomials $m_{(e,d],F}$ and $m_{X_i,F}$. Crucial for our approaches is this can be done efficiently, which will be demonstrated in this section. First, we show how to express the polynomials by other polynomials. A proof can be found in Appendix A.

Lemma 2. *The minimal polynomials are*

$$m_{(e,d],F} = \frac{m_F}{\gcd(m_e, m_F)} \quad \textit{and} \quad m_{X_i,F} = \gcd(m_F, m_{X_i,d}).$$

Now we argue why the expressions on the right side of the equations can be computed efficiently. Let $E := \sum_{i=0}^{e} \binom{n}{i}$ and $D := \sum_{i=0}^{d} \binom{n}{i}$. m_e can be constructed by the method explained in [12, Section 6] with complexity $E[n(\log_2 n)^2 + (\log_2 E)^3]$. m_F can be either computed as described in [9] using Berlekamp-Massey or, in certain cases, with the faster method from [3]. In any case, the complexity is at most the complexity for the Berlekamp-Massey algorithm, which is at most $O(D^2)$. The complexity for gcd and division of two polynomials of degree $\leq D$ over $\mathbb{F}_2$ are $O(M(D)log(D))$ and $O(M(D))$ respectively [1]. Hereby $M(D)$ denotes the complexity of computing the product of two polynomials of degree D which is $O(D \log D \log\log D)$ [18]. Because of $E \leq D$, the overall complexity of computing $m_{(e,d],F}$ is

$$O(D^2 + (D \log D \log\log D)(1 + \log D)).$$

For example, for $n = 128$ and $d = 4$ (as in the case of E_0), the complexity is $\approx 2^{47}$ which is negligible compared to the attack complexity of $\approx 2^{54}$ [9].

In the case of the divide-and-conquer attack, the computation of the minimal polynomial $m_{X_i,F}$ is even easier. First, the algorithm from [12, Section 6] can be easily adapted to compute m_{X_i} (just choose appropriate Ψ). Similarly to above, we can argue that the complexity for computing $m_{X_i,F}$ is in

$$O(D^2 + D \log^2 D \log\log D).$$

Summing up, it is possible to compute efficiently the minimal polynomials described in Section 5.3. As the coefficients of these monomials are exactly the coefficients of the linear combinations used in the precomputation steps, both attacks are feasible with a minimum successive data complexity.

6 Conclusions

Fast algebraic attacks are based on simplifying systems of equations in a precomputation step. Linear combinations of the initial equations are build to reduce the degree. We proposed a new variant where the number of variables is decreased instead.

For both variants, we developed efficient algorithms to find suitable linear combinations where the number of involved equations is minimized.

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. The design and Analysis of Computer Algorithms. Addison-Wesley, Reading MA, 1974.
2. Frederik Armknecht, Matthias Krause: *Algebraic attacks on Combiners with Memory*, Proceedings of Crypto 2003, LNCS 2729, pp. 162-176, Springer, 2003.
3. Frederik Armknecht: *Improving Fast Algebraic Attacks*, Proceedings of Fast Software Encryption 2004, LNCS 3017, pp. 65 - 82, Springer, 2004.
4. Frederik Armknecht: *On the existence of low-degree equations*, Cryptology ePrint Archive: Report 2004/185.
5. Alex Biryukov, Adi Shamir: *Cryptanalytic Time/Memory/Data tradeoffs for Stream Ciphers*, Proceedings of Asiacrypt 2000, LNCS 1976, pp. 1-13, Springer, 2000.
6. Bluetooth SIG, *Specification of the Bluetooth system*, Version 1.1, February 22, 2001. Available at http://www.bluetooth.com/.
7. Nicolas Courtois: *Higher Order Correlation Attacks, XL Algorithm and Cryptanalysis of Toyocrypt*, ICISC 2002, LNCS 2587. An updated version (2002) is available at http://eprint.iacr.org/2002/087/.
8. Nicolas Courtois, Willi Meier: *Algebraic attacks on Stream Ciphers with Linear Feedback*, Eurocrypt 2003, Warsaw, Poland, LNCS 2656, pp. 345-359, Springer, 2003. An extended version is available at http://www.minrnak.org/toyolili.pdf
9. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Proceedings of Crypto '03, LNCS 2729, pp. 177-194, Springer, 2003.
10. Jean-Charles Faugère, Gwenole Ars: *An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases*, 2003. Available at http://www.inria.fr/rrrt/rr-4739.html.
11. Jovan Dj. Golic: *Vectorial Boolean functions and induced algebraic equations*, Cryptology ePrint Archive: Report 2004/225.
12. Philip Hawkes, Gregory G. Rose: *Rewriting Variables: the Complexity of Fast Algebraic Attacks on Stream Ciphers*, Proceeding of Crypto '04, pp. 390-406, LNCS 3152, Springer, 2004. Available at http://eprint.iacr.org/2004/081/.
13. Matthias Krause: *BDD-Based Cryptanalysis of Keystream Generators*; Proceedings of Eurocrypt '02, Springer LNCS 2332, 2002, pp. 222-237.
14. Rudolf Lidl, Harald Niederreiter: *Introduction to finite fields an their applications*, Cambridge University Press, 1994.
15. J. L. Massey: *Shift-register synthesis and BCH decoding*, IEEE Trans. Information Theory, IT-15 (1969), pp. 122-127, 1969.
16. Willi Meier, Enes Pasalic, Claude Carlet: *Algebraic attacks and decomposition of Boolean functions*, Proceeding of Eurocrypt 2004, LNCS 3027, pp. 474-491, Springer, 2004.
17. Willi Meier, Othmar Staffelbach: *Fast Correlation Attacks on certain Stream Ciphers*, Journal of Cryptology, pp. 159-176, 1989.
18. A. Schoenhage: *Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2*, Acta Informatica 7 (1977), pp. 395-398, 1977.
19. L. Simpson, E. Dawson, J. Golic and W. Millan: *LILI Keystream Generator*, Proceeding of SAC '00, pp. 248-261, LNCS 2012, Springer, 2000. See www.isrc.qut.edu.au/lili/.
20. Volker Strassen: *Gaussian Elimination is Not Optimal*; Numerische Mathematik, vol 13, pp 354-356, 1969.

A Proofs

A.1 Proof of Theorem 2

Proof. As L is similar to L^T, there exists a unique matrix $S \in GL_n(\mathbb{F}_2)$ with $L^T \cdot S = S \cdot L$. This implies the existence of n $\mathbb{F}_2$-linear functions $s_0, \ldots, s_{n-1}$ with $(s_0(X), \ldots, s_{n-1}(X)) = S \cdot X$. We define $\varphi_L(X) := \sum_{i=0}^{n-1} s_i(X) \cdot \omega^i$. As the functions s_i are linearly independent and $1, \omega, \ldots, \omega^{n-1}$ forms a basis of $\mathbb{F}_{2^n}$, φ_L is clearly a bijection. As φ_L is the sum of linear functions s_i, φ_L is linear. What is left is to show that $\varphi_L(L^i \cdot X) = \omega^i \cdot \varphi_L(X)$. To do so we define the vector $\Omega = (1, \omega, \ldots, \omega^{n-1})^T$. Then it is $\varphi_L(X) = \langle \Omega, S \cdot X \rangle$ where $\langle ., . \rangle$ denotes the usual vector product. As ω is a root of $p(x)$, it is $\omega^n = \bigoplus_{i=0}^{n-1} a_i \omega^i$. This implies

$$\begin{aligned}
\omega \cdot \varphi_L(X) &= \bigoplus_{i=0}^{n-1} s_i(X) \cdot \omega^{i+1} = \bigoplus_{i=0}^{n-1} s_i(X) \cdot \omega^{i+1} \oplus s_{n-1}(X) \cdot \bigoplus_{i=0}^{n-1} a_i \omega^i \\
&= a_0 s_{n-1}(X) \oplus (s_0(X) \oplus a_1 s_{n-1}(X))\omega \oplus \ldots \\
&\quad \oplus (s_{n-2}(X) \oplus a_{n-1} s_{n-1}(\overrightarrow{X}))\omega^{n-1} \\
&= \langle \Omega, (L^T \cdot S) \cdot X \rangle = \langle \Omega, (S \cdot L) \cdot X \rangle = \langle \Omega, S \cdot (L \cdot X) \rangle = \varphi_L(L \cdot X).
\end{aligned}$$

The rest follows by induction.

A.2 Proof of Theorem 3

Proof. Let $F(X_1, \ldots, X_m) : \mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_m} \to \mathbb{F}_2$ with $X_i = (x_0^{(i)}, \ldots, x_{n_i-1}^{(i)})$. we use the $\mathbb{F}_2$-linear bijections φ_{L_i} from Theorem 2 to set $\tilde{X}_i := \varphi_{L_i}(X_i)$. As φ_{L_i} and the projection are linear, there exists a linear function $\ell_j^{(i)}$ with $x_j^{(i)} = \ell_j^{(i)}(\tilde{X}_i)$. By the definition of φ_{L_i}, it is $x_j^{(i)} = \ell_0^{(i)}(\omega_i^j \cdot \tilde{X}_i)$. Observe that $\omega_1, \ldots, \omega_m$ depend only on the feedback polynomials $p_1, \ldots, p_m$ and are thus independent of the choice of F. This defines a function $\tilde{F}(\tilde{X}_1, \ldots, \tilde{X}_m) : \mathbb{F}_{2^{n_1}} \times \ldots \times \mathbb{F}_{2^{n_m}} \to \mathbb{F}_2$ by

$$\tilde{F} := F(\ell_0^{(1)}(\tilde{X}_1), \ldots, \ell_0^{(1)}(\omega_1^{n_1-1}\tilde{X}_1), \ldots, \ell_0^{(m)}(\tilde{X}_m), \ldots, \ell_0^{(m)}(\omega_m^{n_m-1}\tilde{X}_m)).$$

We use the fact $\mathbb{F}_{2^{n_i}} \subseteq \mathbb{F}_{2^{n'}}$ for $1 \le i \le m$ to extend the domain from $\tilde{F}$ to $(\mathbb{F}_{2^{n'}})^m$:

$$H(Y_1, \ldots, Y_m) := F(\ell_0^{(1)}(Y_1), \ldots, \ell_0^{(1)}(\omega_1^{n_1-1}Y_1), \ldots, \ell_0^{(m)}(Y_m), \ldots, \ell_0^{(m)}(\omega_m^{n_m-1}Y_m)).$$

Let $\Phi(F) := H$. If $\Phi(F_1) = \Phi(F_2)$, this implies $\tilde{F_1} = \tilde{F_2}$ and hence $F_1 = F_2$. Thus, Φ is an injection.

It is easy to see that Φ is linear, i.e., $\Phi(F_1 + F_2) = \Phi(F_1) + \Phi(F_2)$, and that $\Phi(F_1 \cdot F_2) = \Phi(F_1) \cdot \Phi(F_2)$. Therefore, for showing the second claim it suffices to restrict on the case $m = 1$ and F a monomial. We set $m := 1$, $n := n_1$, $\omega := \omega_1$, $Y := Y_1$ and $F(X) = F(x_0, \ldots, x_{n-1}) := \prod_{i \in I} x_i$ for $I := \{i_1, \ldots, i_d\} \subseteq$

$\{0,\ldots,n-1\}$. Obviously, it is $\deg(F)=d$. As the function ℓ_0 is $\mathbb{F}_2$-linear, there exist elements $\lambda_0,\ldots,\lambda_{n-1}\in\mathbb{F}_{2^n}$ such that $\ell_0(\tilde{X})=\sum_{j=0}^{n-1}\lambda_j\tilde{X}^{2^j}$. It is

$$\begin{aligned}\Phi(F(X)) &= \prod_{i\in I}\ell_0(\omega^i Y)=\prod_{i\in I}(\sum_{j=0}^{n-1}\lambda_j\omega^{2^j i}Y^{2^j}))\\ &= \sum_{j_1,\ldots,j_d=0}^{n-1}\omega^{\sum_{k=1}^d i_k 2^{j_k}}\cdot\prod_{k=1}^{d}\lambda_{j_k}\cdot Y^{\sum_{k=1}^d 2^{j_k}}=\sum_{\substack{0\le\alpha\le 2^n-1\\ wt(\alpha)\le d}}c_\alpha Y^\alpha\end{aligned}$$

This shows that $\max\{\sum_{i=1}^m wt(\alpha_i)\,|\,0\le\alpha_i\le 2^{n'}-1, c_{\alpha_1,\ldots,\alpha_m}\ne 0\}\le\deg(F)$.

Now we consider $\Phi(F)=\sum_\alpha c_\alpha Y^\alpha$. As Φ is an injection, there exists a unique $\tilde{F}(\tilde{X})=\sum_\alpha c_\alpha\tilde{X}^\alpha$. By definition, it is $\tilde{X}=\varphi_L(X)$. We use this to identify the sets $\{\mathbb{F}_{2^n}\to\mathbb{F}_{2^n}\}$ and $\{\mathbb{F}_2^n\to\mathbb{F}_{2^n}\}$. From Theorem 2, we known that φ_L is linear and thus coefficients $\mu_0,\ldots,\mu_{n-1}\in\mathbb{F}_{2^n}$ exist with $\tilde{X}=\varphi_L(X)=\varphi_L(x_0,\ldots,x_{n-1})=\sum_{i=0}^{n-1}\mu_i\cdot x_i$. As all fields have characteristic 2 and $x_i\in\mathbb{F}_2$, it is $\tilde{X}^{2^j}=\varphi_L^{2^j}(x_0,\ldots,x_{n-1})=\sum_{i=0}^{n-1}\mu_i^{2^j}\cdot x_i$ for all j. This implies for all $\alpha=2^{\alpha_1}+\ldots+2^{\alpha_d}$ that

$$\tilde{X}^\alpha=\prod_{k=1}^d\tilde{X}^{2^{\alpha_k}}=\prod_{k=1}^d\underbrace{(\sum_{i=0}^{n-1}\mu_i^{2^{\alpha_k}}\cdot x_i)}_{\text{linear in }(x_0,\ldots,x_{n-1})}=P_\alpha(x_0,\ldots,x_{n-1})\in\mathbb{F}_{2^n}[x_0,\ldots,x_{n-1}]$$

with $\deg P_\alpha\le d=wt(\alpha)$. For $\tilde{F}=\sum_\alpha c_\alpha\tilde{X}^\alpha$ this gives $P(x_0,\ldots,x_{n-1}):=\sum_\alpha P_\alpha(x_0,\ldots,x_{n-1})$.

For all $(x_0,\ldots,x_{n-1})\in\mathbb{F}_2^n$, it is $P(x_0,\ldots,x_{n-1})=F(x_0,\ldots,x_{n-1})\in\mathbb{F}_2$ and therefore $P=F$ and $\deg F=\deg P\le\max\{wt(\alpha)\,|\,c_\alpha\ne 0\}$.

A.3 Proof of Lemma 2

Proof. First, we observe that $R_{e,F}\cup(R_e\cap R_F)=R_F$. Furtheron, we claim that $m_{e,F}$ and $m':=m_{(R_e\cap R_F)}$ are co-prime. Then Lemma 1 concludes the proof.

What is left is to show that $m_{e,F}$ and m' are co-prime. Assume, that this is not the case. Then there exists an irreducible polynomial $f\not\equiv 1$ which divides $m_{e,F}$ and m'. As $m_{e,F}$ and m' are minimal, there exists $\omega^\alpha\in R_{e,F}$ and $\omega^\beta\in R_e\cap R_F$ with $f(\omega^\alpha)=f(\omega^\beta)=0$ and $wt(\beta)\le e<wt(\alpha)$. Due to Theorem 1, this implies $\beta=2^k\cdot\alpha \mod (2^{n'}-1)$ and therefore $e<wt(\alpha)=wt(2^k\alpha \mod (2^{n'}-1))=wt(\beta)\le e$ which is a contradiction.

The second claim is true because of $\Omega_{X_i,F}=\Omega_F\cap\Omega_{X_i,d}$ and Lemma 1.

A.4 Proof of Theorem 4

Proof. Let $p(x)=\sum_{i=0}^T\lambda_i x^i$ be an arbitrary polynomial. As Φ_L is linear, we have for all $t\ge 0$:

$$\Phi_L(\bigoplus_{i=0}^{T} \lambda_i F(L^{t+i} \cdot (X_1, \dots, X_m))) = \bigoplus_{i=0}^{T} \lambda_i \Phi_L(F(L^{t+i} \cdot (X_1, \dots, X_m)))$$
$$= \bigoplus_{i=0}^{T} \lambda_i \left(\bigoplus_{\alpha} c_\alpha \omega^{(t+i)\cdot\alpha} Y^\alpha \right) = \sum_\alpha \underbrace{\omega^{\alpha \cdot i}}_{\neq 0} c_\alpha \underbrace{(\bigoplus_{i=0}^{T} \lambda_i (\omega^\alpha)^t Y^\alpha}_{=p(\omega^\alpha)}$$

Let $G := \bigoplus_{i=0}^{T} \lambda_i F_{t+i}$. By Theorem 3, it is

$$\deg(\bigoplus_{i=0}^{T} \lambda_i F(L^{t+i} \cdot (X_1, \dots, X_m)) = \max\{wt(\alpha) \,|\, c_\alpha p(\omega^\alpha) \neq 0\}.$$

This shows (8). The last claim follows from the fact that $\Omega_{(e,d],F} \subseteq \Omega_F$.

B A Toy Example for $m_{(e,d],F} \neq m_F$

The general situation is

$$0 = \underbrace{F_t}_{\deg = d} \oplus \underbrace{G_t}_{\deg = e}$$

with $e < d$. Let $e = 1$, $d = 2$ and $F_t := x_t \cdot x_{t+1}$ with x_t being the LFSRs output at clock t. The feedback is defined by $x_{t+2} := x_t \oplus x_{t+1}$. The first outputs of F are:

Clock	F_t
0	$x_0 \cdot x_1$
1	$x_1 \cdot x_2 = x_1 \cdot (x_0 \oplus x_1) = x_1 \oplus x_0 \cdot x_1$
2	$x_2 \cdot x_3 = (x_0 \oplus x_1) \cdot x_0 = x_0 \oplus x_0 \cdot x_1$
3	$x_3 \cdot x_4 = x_0 \cdot x_1$
...	...

It is easy to see that

$$m_F = 1 + x^3 \Rightarrow F_t \oplus F_{t+3} = 0$$
$$m_{(1,2],F} = 1 + x \Rightarrow \deg(F_t \oplus F_{t+1}) = 1 = e$$

In this case, the successive data complexity can be reduced from 3 (using m_F) to 1 (using $m_{(e,d],F}$).

C The Minimum Successive Data Complexity in the Case of E_0

The E_0 keystream generator which is part of the Bluetooth standard, uses four LFSRs A, B, C and D of lengths $n_a = 25$, $n_b = 31$, $n_a = 33$, $n_a = 39$, We denote their outputs at clock t by a_t, b_t, c_t and d_t and use the equation derived in [2]. The according F is $F = \sigma_t^4 \oplus \sigma_t^2 \cdot \sigma_{t+1}^2$ where $\sigma_t^4 = a_t b_t c_t d_t$ and $\sigma_t^2 = a_t b_t \oplus a_t c_t \oplus a_t d_t \oplus b_t c_t \oplus b_t d_t \oplus c_t d_t$. By Lemma 2, it is $\deg(m_{(e,d],F}) = \deg(m_F) - \deg(gcd(m_e, m_F))$. It was estimated in [3] that $\deg(m_F) = 8,822,188$. Furtheron, an upper bound for $\deg(gcd(m_e, m_F))$ is the number of monomials of degree ≤ 3, which can occur in the system of equations given by F. We calculated that this number is $\approx 326,080$. This shows that the number of successive bits required for one equation of degree 3 is $\approx 8,496,108$.

Equivalent Keys in HFE, C*, and Variations

Christopher Wolf and Bart Preneel

K.U.Leuven, ESAT-COSIC,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{Christopher.Wolf, Bart.Preneel}@esat.kuleuven.ac.be,
chris@Christopher-Wolf.de
http://www.esat.kuleuven.ac.be/cosic/

Abstract. In this article, we investigate the question of equivalent keys for two $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic public key schemes HFE and C^{*--} and improve over a previously known result, which appeared at PKC 2005. Moreover, we show a new non-trivial extension of these results to the classes HFE-, HFEv, HFEv-, and C^{*--}, which are cryptographically stronger variants of the original HFE and C^* schemes. In particular, we are able to reduce the size of the private — and hence the public — key space by at least one order of magnitude and several orders of magnitude on average. While the results are of independent interest themselves as they broaden our understanding of $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic schemes, we also see applications both in cryptanalysis and in memory efficient implementations.

Keywords: Multivariate Quadratic Equations, Public Key signature, Hidden Field Equations, HFE, HFE-, HFEv, HFEv-, C^*, C^{*--}

1 Introduction

In the last 15 years, several schemes based on the problem of $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic equations have been proposed. The most important ones certainly are C^* [9] and Hidden Field Equations (HFE, [13]) plus their variations C^{*--}, HFE-, HFEv, and HFEv- [7,12,13]. Both have been used to construct signature schemes, namely C^{*--} in Sflash [3], and HFEv- in Quartz [2]. As for all systems based on $\mathcal{MQ}$-equations, the public key has the form

$$p_i(x_1, \ldots, x_n) := \sum_{1 \le j \le k \le n} \gamma_{i,j,k} x_j x_k + \sum_{j=1}^{n} \beta_{i,j} x_j + \alpha_i \,,$$

for $1 \le i \le m; 1 \le j \le k \le n$ and $\alpha_i, \beta_{i,j}, \gamma_{i,j,k} \in \mathbb{F}$ (constant, linear, and quadratic terms). We write the set of all such equations as $\mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m)$. Moreover, the private key consists of the triple $(S, \mathcal{P}', T)$ where $S \in \mathrm{Aff}(\mathbb{F}^n), T \in \mathrm{Aff}(\mathbb{F}^m)$ are affine transformations (cf Sect. 2.2) and $\mathcal{P}' \in \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m)$ is a polynomial-vector $\mathcal{P}' := (p'_1, \ldots, p'_m)$ with m components; each component is a polynomial in n variables $x'_1, \ldots, x'_n$. Throughout this paper, we will denote components of this private vector $\mathcal{P}'$ by a prime $'$. In contrast to the public polynomial vector $\mathcal{P} \in \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m)$, the private polynomial vector $\mathcal{P}'$ does allow an efficient computation of $x'_1, \dots, x'_n$ for given $y'_1, \dots, y'_m$. Hence, the goal of $\mathcal{MQ}$-schemes is that this inversion should be hard if the public key $\mathcal{P}$ alone is given. The main difference between $\mathcal{MQ}$-schemes lies in their special construction of the central equations $\mathcal{P}'$ and consequently the trapdoor they embed into a specific class of $\mathcal{MQ}$-problems.

In this paper, we investigate the question of equivalent keys for selected $\mathcal{MQ}$-schemes. Due to space limitations, we concentrate on HFE, HFE-, HFEv, HFEv-, C*, and C^{*--}. As outlined above, they are quite important as they have been used in constructions submitted to the NESSIE project [10]. However, we want to point out that the techniques outlined here are quite general and can also be applied to other schemes. The first paper on the topic of equivalent keys is [19]. In this paper, we introduce the Frobenius sustainer and are hence able to improve over the results from [19]. Moreover, this paper is the first to deal with variations of $\mathcal{MQ}$-schemes, cf [20] for the terminology of $\mathcal{MQ}$-trapdoors. To this aim, we needed to develop the reduction sustainer, as we would not have been able to deal with the HFE- and the C^{*--} modification otherwise.

This paper is outlined as follows: after this general introduction, we move on to the necessary mathematical background in Sect. 2. This includes particularly a definition of the term *equivalent keys.* In Sect. 3, we concentrate on a subclass of affine transformations, denoted *sustaining transformations*, which can be used to generate equivalent keys. These transformations are applied to different variations of $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic equations in Sect. 4. This paper concludes with Sect. 5, cf [19] for results on Unbalanced Oil and Vinegar schemes (UOV). A general overview of $\mathcal{MQ}$-schemes can be found in [20].

2 Mathematical Background

In this section, we outline some observations useful in the remainder of this paper.

2.1 Basic Definitions

We start with a formal definition of the term "equivalent private keys":

Definition 1. *We call two private keys*

$$(T, \mathcal{P}', S), (\tilde{T}, \tilde{\mathcal{P}}', \tilde{S}) \in \mathit{Aff}(\mathbb{F}^m) \times \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m) \times \mathit{Aff}(\mathbb{F}^n)$$

"equivalent" if they lead to the same public key, i.e., if we have

$$T \circ \mathcal{P}' \circ S = \mathcal{P} = \tilde{T} \circ \tilde{\mathcal{P}}' \circ \tilde{S}\,.$$

In order to find equivalent keys, we consider the following transformations:

Definition 2. *Let $(S, \mathcal{P}', T) \in \mathit{Aff}(\mathbb{F}^m) \times \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^m) \times \mathit{Aff}(\mathbb{F}^n)$, and $\sigma, \sigma^{-1} \in \mathit{Aff}(\mathbb{F}^n)$ plus $\tau, \tau^{-1} \in \mathit{Aff}(\mathbb{F}^m)$. Moreover, let*

$$\mathcal{P} = T \circ \tau^{-1} \circ \tau \circ \mathcal{P}' \circ \sigma \circ \sigma^{-1} \circ S \tag{1}$$

We call the pair $(\sigma, \tau) \in \mathit{Aff}(\mathbb{F}^n) \times \mathit{Aff}(\mathbb{F}^m)$ "sustaining transformations" for an $\mathcal{MQ}$-system if the "shape" of $\mathcal{P}'$ is invariant under the transformations σ and τ. For short, we write $(\sigma, \tau) \bullet (S, \mathcal{P}', T)$ for (1) and (σ, τ) sustaining transformations.

Remark 1. In the above definition, the meaning of "shape" is still open. In fact, its meaning has to be defined for each $\mathcal{MQ}$-system individually. For example, in HFE (cf Sect. 4.1), it is the bounding degree $d \in \mathbb{N}$ of the polynomial $P'(X')$. In the case of C*, the "shape" is the fact that we have a single monomial with factor 1 as the central equation (cf Sect. 4.2). However, for σ, τ sustaining transformations, we are now able to produce equivalent keys for a given private key by $(\sigma, \tau) \bullet (S, \mathcal{P}', T)$. A trivial example of sustaining transformations is the identity transformation, *i.e.*, to set $\sigma = \tau = id$.

Lemma 1. *Let (σ, τ) be sustaining transformation. If $G := (\sigma, \circ)$ and $H := (\tau, \circ)$ form a subgroup of the affine transformations, they produce equivalence relations within the private key space.*

Proof. We start with a proof of this statement for $G := (\sigma, \circ)$. First, we have reflexivity as the identity transformation is contained in G. Second, we have symmetry as subgroups are closed under inversion. Third, we also have transitivity as subgroups are closed under composition. Therefore, the group G partitions the private key space into equivalence classes. The proof for $H := (\tau, \circ)$ is analogous.

Remark 2. We want to point out that the above proof does not use special properties of sustaining transformations, but the fact that these are a subgroup of the group of affine transformations. Hence, the proof does not depend on the term "shape" and is therefore valid even if the latter is not rigorously defined yet. In any case, instead of proving that sustaining transformations form a subgroup of the affine transformations, we can also consider normal forms of private keys. As we see below, normal forms have some advantages to avoid double counts in the private key space.

After these initial observations over equivalent keys, we concentrate on bijections between ground fields and their extension fields as both HFE and C* use an extension field to define their central equations $\mathcal{P}'$. Let $\mathbb{F}$ be a finite field with $q := |\mathbb{F}|$ elements. Using a polynomial $i(t) \in \mathbb{F}[t]$, irreducible over $\mathbb{F}$, we generate an extension field $\mathbb{E} := \mathbb{F}[t]/i(t)$ of dimension n. This means we view elements of $\mathbb{E}$ as polynomials in t of degree less than n. Addition and multiplication are defined as for polynomials modulo $i(t)$. In addition, we can view elements from $\mathbb{E}$ as vectors over the vector-space $\mathbb{F}^n$. We will therefore view elements $a \in \mathbb{E}$ and $b \in \mathbb{F}^n$ as

$$a := \alpha_{n-1} t^{n-1} + \ldots + \alpha_1 t + \alpha_0 \text{ and } b := (\beta_1, \ldots, \beta_n),$$

for $\alpha_{i-1}, \beta_i \in \mathbb{F}$ with $1 \leq i \leq n$. Moreover, we define the *canonical bijection* between $\mathbb{E}$ and $\mathbb{F}^n$ by identifying the coefficients $\alpha_{i-1} \leftrightarrow \beta_i$. We use both this bijection $\phi : \mathbb{E} \rightarrow \mathbb{F}^n$ and its inverse $\phi^{-1} : \mathbb{F}^n \rightarrow \mathbb{E}$.

2.2 Affine Transformations

In the context of affine transformations, the following lemma proves useful:

Lemma 2. *Let $\mathbb{F}$ be a finite field with $q := |\mathbb{F}|$ elements. Then we have $\prod_{i=0}^{n-1} q^n - q^i$ invertible $(n \times n)$-matrices over $\mathbb{F}$.*

Next, we recall some basic properties of affine transformations over the finite fields $\mathbb{F}$ and $\mathbb{E}$.

Definition 3. *Let $M_S \in \mathbb{F}^{n \times n}$ be an invertible $(n \times n)$ matrix and $v_s \in \mathbb{F}^n$ a vector and let $S(x) := M_S x + v_s$. We call this the "matrix representation" of the affine transformation S.*

Definition 4. *Moreover, let $s_1, \ldots, s_n$ be n polynomials of degree 1 at most over $\mathbb{F}$, i.e., $s_i(x_1, \ldots, x_n) := \beta_{i,1}x_1 + \ldots + \beta_{i,n}x_n + \alpha_i$ with $1 \leq i, j \leq n$ and $\alpha_i, \beta_{i,j} \in \mathbb{F}$. Let $S(x) := (s_1(x), \ldots, s_n(x))$ for $x := (x_1, \ldots, x_n)$ as a vector over $\mathbb{F}^n$. We call this the "multivariate representation" of the affine transformation S.*

Remark 3. The multivariate and the matrix representation of an affine transformation S are interchangeable. We only need to set the corresponding coefficients to the same values: $(M_S)_{i,j} \leftrightarrow \beta_{i,j}$ and $(v_S)_i \leftrightarrow \alpha_i$ for $1 \leq i, j \leq n$.

In addition, we can also use the "univariate representation" over the extension field $\mathbb{E}$ of the transformation S.

Definition 5. *Let $0 \leq i < n$ and $A, B_i \in \mathbb{E}$. Moreover, let the polynomial $S(X) := \sum_{i=0}^{n-1} B_i X^{q^i} + A$ be an affine transformation. We call this the "univariate representation" of the affine transformation $S(X)$.*

Lemma 3. *An affine transformation in univariate representation can be transfered efficiently in multivariate representation and vice versa.*

Proof. This lemma follows from [8, Lemmata 3.1 and 3.2] by a simple extension from the linear to the affine case.

3 Sustaining Transformations

In this section, we discuss several examples for sustaining transformations. In addition, we will consider their effect on the central transformation $\mathcal{P}'$. The authors are not convinced that the transformations stated here are the only ones possible but encourage the search for other and maybe more powerful sustaining transformations.

3.1 Additive Sustainer

For $n = m$, let $\sigma(X) := (X + A)$ and $\tau(X) := (X + A')$ for some elements $A, A' \in \mathbb{E}$. Moreover, as long as they keep the shape of the central equations $\mathcal{P}'$ invariant, they form sustaining transformations.

In particular, we are able to change the constant parts $v_s, v_t \in \mathbb{F}^n$ or $V_S, V_T \in \mathbb{E}$ of the two affine transformations $S, T \in \mathrm{Aff}(\mathbb{F}^n)$ to zero, *i.e.*, to obtain a new key $(\hat{S}, \hat{\mathcal{P}}', \hat{T})$ with $\hat{S}, \hat{T} \in \mathrm{Hom}(\mathbb{F}^n)$.

Remark 4. This is a very useful result for cryptanalysis as it allows us to "collect" the constant terms in the central equations $\mathcal{P}'$. For cryptanalytic purposes, we therefore need only to consider the case of linear transformations $S, T \in \mathrm{Hom}(\mathbb{F}^n)$.

The additive sustainer also works if we interpret it over the vector space $\mathbb{F}^n$ rather than the extension field $\mathbb{E}$. In particular, we can also handle the case $n \neq m$ now. However, in this case it may happen that we have $a' \in \mathbb{F}^m$ and consequently $\tau : \mathbb{F}^m \rightarrow \mathbb{F}^m$. Nevertheless, we can still collect all constant terms in the central equations $\mathcal{P}'$.

If we look at the central equations as multivariate polynomials, the additive sustainer will affect the constants α_i and $\beta_{i,j} \in \mathbb{F}$ for $1 \leq i \leq m$ and $1 \leq j \leq n$. A similar observation is true for central equations over the extension field $\mathbb{E}$: in this case, the additive sustainer affects the additive constant $A \in \mathbb{E}$ and the linear factors $B_i \in \mathbb{E}$ for $0 \leq i < n$.

3.2 Big Sustainer

We now consider multiplication in the (big) extension field $\mathbb{E}$, *i.e.*, we have $\sigma(X) := (BX)$ and $\tau(X) := (B'X)$ for $B, B' \in \mathbb{E}^*$. Again, we obtain a sustaining transformation if this operation does not modify the shape of the central equations as $(BX), (B'X) \in \mathrm{Aff}(\mathbb{F}^n)$.

The big sustainer is useful if we consider schemes defined over extension fields as it does not affect the overall degree of the central equations over this extension field.

3.3 Small Sustainer

We now consider multiplications over the (small) ground field $\mathbb{F}$, *i.e.*, we have $\sigma(x) := \mathit{Diag}(b_1, \ldots, b_n)x$ and $\tau(x) := \mathit{Diag}(b'_1, \ldots, b'_m)x$ for the coefficients $b_1, \ldots, b_n, b'_1, \ldots, b'_m \in \mathbb{F}^*$ and $\mathit{Diag}(b)$ the diagonal matrix on a vector $b \in \mathbb{F}^n$ and $b' \in \mathbb{F}^m$, respectively.

In contrast to the big sustainer, the small sustainer is useful if we consider schemes which define the central equations over the ground field $\mathbb{F}$ as it only introduces a scalar factor in the polynomials $(p'_1, \ldots, p'_m)$.

3.4 Permutation Sustainer

For the transformation σ, this sustainer permutes input-variables of the central equations whilefor the transformation τ, it permutes the polynomials of the central equations themselves. As each permutation has a corresponding, invertible permutation-matrix, both $\sigma \in S_n$ and $\tau \in S_m$ are also affine transformations. The effect of the central equations is limited to a permutation of these equations and their input variables, respectively.

3.5 Gauss Sustainer

Here, we consider Gauss operations on matrices, *i.e.*, row and column permutations, multiplication of rows and columns by scalars from the ground field $\mathbb{F}$, and the addition of two rows/columns. As all these operations can be performed by invertible matrices; they form a subgroup of the affine transformations and are hence a candidate for a sustaining transformation.

The effect of the Gauss Sustainer is similar to the permutation sustainer and the small sustainer. In addition, it allows the addition of multivariate quadratic polynomials. This will not affect the shape of some $\mathcal{MQ}$-schemes.

The sustainers given so far have been already outlined in [19]. To the knowledge of the authors, the following sustainers are new and to the knowledge to the authors have not been considered previously in the literature.

3.6 Frobenius Sustainer

Definition 6. *Let $\mathbb{F}$ be a finite field with $q := |\mathbb{F}|$ elements and $\mathbb{E}$ its n-dimensional extension. Moreover, let $H := \{i \in \mathbb{Z} : 0 \leq i < n\}$. For $a, b \in H$ we call $\sigma(X) := X^{q^a}$ and $\tau(X) := X^{q^b}$* Frobenius transformations.

Obviously, Frobenius transformations are linear transformations with respect to $\mathbb{F}$. The following lemma establishes that they also form a group:

Lemma 4. *Frobenius transformations are a subgroup in Hom$(\mathbb{F}^n)$.*

Proof. First, Frobenius transformations are linear transformations, so associativity is inherited from them. Second, the set H from Def. 6 is not empty for any given $\mathbb{F}$ and $n \in \mathbb{N}$. Hence, the corresponding set of Frobenius transformations is not empty either. So all left to show is that for any given Frobenius transformations σ, τ, the composition $\sigma \circ \tau^{-1}$ is also a Frobenius transformation.

Let $\sigma(X) := X^{q^a}$ and $\tau(X) := X^{q^b}$ for some $a, b \in H$. Working in the multiplicative group $\mathbb{E}^*$ we observe that we need $q^b \cdot B' \equiv 1 \pmod{q^n - 1}$ for B' to obtain the inverse function of τ. We notice that $B' := q^{b'}$ for $b' := n - b \pmod{n}$ yields the required and moreover $\tau^{-1} := X^{q^{b'}}$ is a Frobenius transformation as $b' \in H$.

So we can write $\sigma(X) \circ \tau^{-1}(X) = X^{q^{a+b'}}$. If $a + b' < n$ we are done. Otherwise $n \leq a + b' < 2n$, so we can write $q^{a+b'} = q^{n+s}$ for some $s \in H$. Again, working in the multiplicative group E^* yields $q^{n+s} \equiv q^s \pmod{q^n - 1}$ and hence, we established that $\sigma \circ \tau^{-1}$ is also a Frobenius transformation. This completes the proof that all Frobenius transformations form a group.

Frobenius transformations usually change the degree of the central equation $\mathcal{P}'$. But taking $\tau := \sigma^{-1}$ cancels this effect and hence preserves the degree of $\mathcal{P}'$.

Therefore, we can speak of a Frobenius sustainer (σ, τ). So there are n Frobenius sustainers for a given extension field $\mathbb{E}$.

It is tempting to extend this result to the case of powers of the characteristic of $\mathbb{F}$. However, this is not possible as $x^{\mathrm{char}\mathbb{F}}$ is not a linear transformation in $\mathbb{F}$ for $q \neq p$.

Remark 5. We want to point out that all six sustainers presented so far form groups and hence partition the private key space into equivalence classes (cf Lemma 1).

3.7 Reduction Sustainer

Reduction sustainers are quite different from the transformations studied so far, because they are applied with a different construction of the trapdoor of $\mathcal{P}$. In this new construction, we define the public key equations as $\mathcal{P} := R \circ T \circ \mathcal{P}' \circ S$ where $R : \mathbb{F}^n \to \mathbb{F}^{n-r}$ denotes a *reduction* or *projection*. In addition, we have $S, T \in \mathrm{Aff}(\mathbb{F}^n)$ and $\mathcal{P}' \in \mathcal{MQ}(\mathbb{F}^n, \mathbb{F}^n)$. Less loosely speaking, we consider the function $R(x_1, \ldots, x_n) := (x_1, \ldots, x_{n-r})$, *i.e.*, we neglect the last r components of the vector $(x_1, \ldots, x_n)$. Although this modification looks rather easy, it proves powerful to defeat a wide class of cryptographic attacks against several $\mathcal{MQ}$-schemes, including HFE and C*, *e.g.*, the attack introduced in [5].

For the corresponding sustainer, we consider the affine transformation T in matrix representation, *i.e.*, we have $T(x) := Mx + v$ for some invertible matrix $M \in \mathbb{F}^{m \times m}$ and a vector $v \in \mathbb{F}^m$. We observe that any change in the last r columns of M or v does not affect the result of R (and hence $\mathcal{P}$). Hence, we can choose these last r columns without affecting the public key. Inspecting Lemma 2, we see that this gives us a total of

$$q^r \prod_{i=n-r-1}^{n-1} (q^n - q^i)$$

choices for v and M, respectively, that do not affect the public key equations $\mathcal{P}$.

When applying the reduction sustainer together with other sustainers, we have to make sure that we do not count the same transformation twice, cf the corresponding proofs.

4 Application to $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Schemes

In this section, we show how to apply the sustainers from the previous section to several $\mathcal{MQ}$-schemes. Due to space limitations in this paper, we will only outline some central properties of each scheme and sketch the corresponding proofs.

4.1 Hidden Field Equations

The Hidden Field Equations (HFE) have been proposed by Patarin [13].

Definition 7. *Let $\mathbb{E}$ be a finite field and P(X) a polynomial over $\mathbb{E}$. For*

$$P(X) := \sum_{\substack{0 \le i,j \le d \\ q^i + q^j \le d}} C_{i,j} X^{q^i + q^j} + \sum_{\substack{0 \le k \le d \\ q^k \le d}} B_k X^{q^k} + A$$

$$\text{where} \begin{cases} C_{i,j} X^{q^i+q^j} & \text{for } C_{i,j} \in \mathbb{E} \text{ are the quadratic terms,} \\ B_k X^{q^k} & \text{for } B_k \in \mathbb{E} \text{ are the linear terms, and} \\ A & \text{for } A \in \mathbb{E} \text{ is the constant term} \end{cases}$$

and a degree $d \in \mathbb{N}$, we say the central equations $\mathcal{P}'$ are in HFE-shape.

Due to the special form of $P(X)$, we can express it as a $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic equation $\mathcal{P}'$ over $\mathbb{F}$, cf [13]. Moreover, as the degree of the polynomial P is bounded by d, this allows efficient inversion of the equation $P(X) = Y$ for given $Y \in \mathbb{E}$. So the *shape* of HFE is in particular this degree d of the private polynomial P. Moreover, we observe that there are no restrictions on its coefficients $C_{i,j}, B_k, A \in \mathbb{E}$ for $i, j, k \in \mathbb{N}$ and $q^i, q^i + q^j \le d$. Hence, we can apply both the additive and the big sustainer (cf sect. 3.1 and 3.2) without changing the shape of this central equation.

Theorem 1. *For $K := (S, P, T) \in Aff(\mathbb{F}^n) \times \mathbb{E}[X] \times Aff(\mathbb{F}^n)$ a private key in HFE, we have*

$$n.q^{2n}(q^n - 1)^2$$

equivalent keys.

Proof. To prove this lemma, we consider normal forms of private keys: let $\tilde{S} \in \text{Aff}(\mathbb{F}^n)$ being the affine transformation we start with. First we compute $\hat{S}(X) := \tilde{S}(X) - \tilde{S}(0)$, *i.e.*, we apply the additive sustainer. Obviously, we have $\hat{S}(0) = 0$ after this transformation and hence a special fix-point. Second we define $\overline{S}(X) := \hat{S}(X).\hat{S}(1)^{-1}$, *i.e.*, we apply the big sustainer. As the transformation $\hat{S} : \mathbb{E} \to \mathbb{E}$ is a bijection and we have $\hat{S}(0) = 0$, we know that $\hat{S}(1)$ must be non-zero. Hence, we have $\overline{S}(1) = 1$, *i.e.*, we add a new fix-point but still keep the old fix-point as we have $\overline{S}(0) = \hat{S}(0) = 0$. Similar we can compute an affine transformation $\overline{T}(X)$ with $\overline{T}(0) = 0$ and $\overline{T}(1) = 1$ as a normal form of the affine transformation $\tilde{T} \in \text{Aff}(\mathbb{F}^n)$. Note that both the additive sustainer and the big sustainer keep the degree of the central polynomial $P(X)$ so we can apply both sustainers on both sides without changing the "shape" of $P(X)$.

Applying the Frobenius sustainer is a little more technical. First we observe that this sustainer keeps the fix-points $\overline{S}(0) = \overline{T}(0) = 0$ and $\overline{S}(1) = \overline{T}(1) = 1$ so we are sure we still deal with equivalence classes, *i.e.*, each given private key has a unique normal form, even with the Frobenius sustainer applied. To this aim we pick an element $C \in \mathbb{E} \backslash \{0, 1\}$ with $g := \overline{S}(C)$ is a generator of $\mathbb{E}^*$, *i.e.*, we have $\mathbb{E}^* = \{g^i \mid 0 \le i < q^n\}$. As $\mathbb{E}$ is a finite field we know that such an element g exists. Given that $\overline{S}$ is injective we know that we can find the corresponding $C \in \mathbb{E} \backslash \{0, 1\}$. Now we compute $g_i := \overline{S}(C)^{q^i}$ for $0 \le i < n$. Using any total ordering "<", we obtain $g_c := \min\{g_0, \dots, g_{n-1}\}$ for some $c \in \mathbb{N}$ as the smallest element of this set. One example of such a total ordering would be to use a bijection between the sets $\mathbb{E} \leftrightarrow \{0, \ldots, q^n - 1\}$ and then exploiting the ordering of the natural numbers to derive an ordering on the elements of the extension field $\mathbb{E}$. Finally, we define $S(X) := [\overline{S}(X)]^{q^c}$ as new affine transformation. To cancel the effect of the Frobenius sustainer, we moreover define $T(X) := [\overline{T}(X)]^{q^{n-c}}$.

Hence, we have now computed a unique normal form for a given private key. Moreover, we can "reverse" these computations and derive an equivalence class of size $n.q^{2n}.(q^n - 1)^2$ this way as we have

$$(BX^{q^c} + A, B'X^{q^{n-c}} + A') \bullet (S, \mathcal{P}', T)$$

for$B, B' \in \mathbb{E}^*, A, A' \in \mathbb{E}$ and $0 \leq c < n$.

Remark 6. To the knowledge of the authors, the additive sustainer for HFE has first been reported in [14] and used there for reducing the affine transformations to linear ones. In addition, a weaker version of the above theorem can be found in [19].

For $q = 2$ and $n = 80$, the number of equivalent keys per private key is $\approx 2^{326}$. In comparison, the number of choices for S and T is $\approx 2^{12,056}$. This special choice of parameters has been used in HFE Challenge 1 [13].

HFE- The class HFE- is the original HFE-class with the reduction modification (cf Sect. 3.7).

Theorem 2. *For $K := (S, P, T) \in Aff(\mathbb{F}^n) \times \mathbb{E}[X] \times Aff(\mathbb{F}^n)$ a private key in HFE and a reduction parameter $r \in \mathbb{N}$ we have*

$$n.q^{2n}(q^n - 1)(q^{n-r} - 1) \prod_{i=n-r-1}^{n-1} (q^n - q^i)$$

equivalent keys and the key-space of HFE- can be reduced by this number.

Proof. This proof uses the same ideas as the proof of Thm. 1 to obtain a normal form of the affine transformation S, *i.e.*, applying the additive sustainer, the big sustainer and the Frobenius sustainer on this side. Hence, we have a reduction by $n.q^n(q^n - 1)$ keys here.

For the affine transformation T, we also have to take the reduction sustainer into account: we use $\tilde{T}(X) : \mathbb{F}^n \to \mathbb{F}^{n-r}$ and fix $\tilde{T}(0) = 0$ by applying the additive sustainer and $\tilde{T}(1) = 1$ by applying the big sustainer, which gives us q^{n-r} and $q^{n-r} - 1$ choices, respectively. To avoid double counting with the reduction sustainer, all computations were performed in $\tilde{\mathbb{E}} := \mathrm{GF}(q^{n-r})$ rather than $\mathbb{E}$. Again, we are able to compute a normal form for a given private key and reverse these computations to obtain the full equivalence class for any given private key in normal form. Moreover, we observe that the resulting transformation $\tilde{T}$ actually allows for $q^r \prod_{i=n-r-1}^{n-1}(q^n - q^i)$ possible choices for the original transformation $T : \mathbb{F}^n \to \mathbb{F}^n$ (reduction sustainer) without affecting the output of

$\tilde{T}$. Hence, there are a total of $q^{n-r}.q^r.(q^{n-r}-1).\prod_{i=n-r-1}^{n-1}(q^n-q^i)$ possibilities for the transformation T without changing the output of the private key triple (S, P', T). Multiplying out the intermediate results for S and T yields the theorem.

For $q = 2, r = 7$ and $n = 107$, the number of equivalent keys for each private key is $\approx 2^{2129}$. In comparison, the number of choices for S and T is $\approx 2^{23,108}$. This special choice of parameters has been used in the repaired version Quartz-7m of Quartz [2,17].

HFEv. The following modification, due to [7], uses a different form for the central equations $\mathcal{P}'$.

Definition 8. *Let $\mathbb{E}$ be a finite field with degree n' over $\mathbb{F}$, the number of vinegar variables $v \in \mathbb{N}$, and P(X) a polynomial over $\mathbb{E}$. Moreover, let $(z_1, \dots, z_v) := s_{n-v+1}(x_1, \dots, x_n), \dots, s_n(x_1, \dots, x_n)$ for s_i the polynomials of $S(x)$ in multivariate representation and $X' := \phi^{-1}(x'_1, \dots, x'_{n'})$, using the canonical bijection $\phi^{-1} : \mathbb{F}^n \to \mathbb{E}$ and $x'_i := s_i(x_1, \dots, x_n)$ for $1 \le i \le n'$ as hidden variables. Then define the central equation as*

$$P'_{z_1,\dots,z_v}(X') := \sum_{\substack{0\le i,j\le d\\ q^i+q^j\le d}} C_{i,j}X'^{q^i+q^j} + \sum_{\substack{0\le k\le d\\ q^k\le d}} B_k(z_1,\dots,z_v)X'^{q^k} + A(z_1,\dots,z_v)$$

$$\textit{where}\begin{cases} C_{i,j}X'^{q^i+q^j} & \textit{for } C_{i,j} \in \mathbb{E} \textit{ are the quadratic terms,}\\ B_k(z_1,\dots,z_v)X'^{q^k} & \textit{for } B_k(z_1,\dots,z_v) \textit{ depending linearly on } z_1,\dots,z_v \textit{ and}\\ A(z_1,\dots,z_v) & \textit{for } A(z_1,\dots,z_v) \textit{ depending quadratically on } z_1,\dots,z_v \end{cases}$$

and a degree $d \in \mathbb{N}$, we say the central equations $\mathcal{P}'$ are in HFEv-shape.

The condition that the $B_k(z_1, \dots, z_v)$ are affine functions (*i.e.*, of degree 1 in the z_i at most) and $A(z_1, \dots, z_v)$ is a quadratic function over $\mathbb{F}$ ensures that the public key is still quadratic over $\mathbb{F}$.

Theorem 3. *For $K := (S, P, T) \in Aff(\mathbb{F}^n) \times \mathbb{E}[X] \times Aff(\mathbb{F}^m)$ a private key in HFEv, $v \in \mathbb{N}$ the number of vinegar variables, $\mathbb{E}$ an n'-dimensional extension of $\mathbb{F}$ where $n' := n - v = m$ we have*

$$n'q^{n+n'}(q^{n'}-1)^2\prod_{i=0}^{v-1}(q^v-q^i)$$

equivalent keys. Hence, the key-space of HFEv can be reduced by this number.

Proof. In contrast to HFE-, the difficulty now lies in the computation of a normal form for the affine transformation S rather than the affine transformation T.

For the latter, we can still apply the big sustainer and the additive sustainer and obtain a total of $q^m.(q^m - 1) = q^{n'}.(q^{n'} - 1)$ equivalent keys for a given transformation T. Moreover, the HFEv modification does not change the "absorbing behaviour" of the central polynomial P and hence, the proof from Thm. 1 is still applicable.

Instead, we have to concentrate on the affine transformation S here. To simplify the following argument, we apply the additive sustainer on S and obtain a linear transformation. This reduces the key-space by q^n. To make sure that we do not count the same linear transformation twice, we consider a normal form for the now (linear) transformation S

$$\begin{pmatrix} E_m & F_v^m \\ G_m^v & I_v \end{pmatrix} \text{ with } E_m \in \mathbb{F}^{m \times m}, F_v^m \in \mathbb{F}^{m \times v}, G_m^v \in \mathbb{F}^{v \times m}$$

In the above definition, we also have I_v the identity matrix in $\mathbb{F}^{v \times v}$. For each invertible matrix M_S, we have a unique matrix

$$\begin{pmatrix} I_m & 0 \\ 0 & H_v \end{pmatrix} \text{ with an invertible matrix } H_v \in \mathbb{F}^{v \times v}.$$

which transfers M_S to the normal form from above. Again, I_m is an identity matrix in $\mathbb{F}^{m \times m}$. This way, we obtain $\prod_{i=0}^{v-1}(q^v - q^i)$ equivalent keys in the "v" modification alone.

For the HFE component over $\mathbb{E}$, we can now apply the big sustainer to S and obtain a factor of $(q^{n'} - 1)$. In addition, we apply the Frobenius sustainer to the HFE component, which yields an additional factor of n'. Note that the Frobenius sustainer can be applied both to S and T, and hence, we can make sure that it cancels out and does not affect the degree of the central polynomial $P_{z_1,\ldots,z_v}(X)$. Again, we can reverse all computations and therefore, obtain equivalence classes of equal size for each given private key in normal form.

For the case $q = 2, v = 7$ and $n = 107$, the number of equivalent keys for each private is $\approx 2^{460}$. In comparison, the number of choices for S and T is $\approx 2^{21,652}$.

HFEv- Here, we combine both the HFEv and the HFE- modification to obtain HFEv-.

Theorem 4. *For $K := (S, P, T) \in Aff(\mathbb{F}^n) \times \mathbb{E}[X] \times Aff(\mathbb{F}^{m+v})$ a private key in HFEv, $v \in \mathbb{N}$ vinegar variables, a reduction parameter $r \in \mathbb{N}$ and $\mathbb{E}$ an n'-dimensional extension of $\mathbb{F}$ where $n' := n - v$ and $n' = m + r$ we have*

$$n' q^r q^{2n'} (q^{n'} - 1)^2 \prod_{i=0}^{v-1} (q^v - q^i) \prod_{i=n-r-1}^{n-1} (q^n - q^i)$$

equivalent keys and the key-space of HFEv can be reduced by this number.

Proof. This proof is a combination of the two cases HFEv and HFE-. Given that the difficulty for the HFE- modification was in the T-transformation while the difficulty of HFEv was in the S-transformation, we can safely combine the known sustainers without any double-counting.

For the case $q = 2, r = 3, v = 4$ and $n = 107, n' := 100$, the number of redundant keys is $\approx 2^{690}$. In comparison, the number of choices for S and T is $\approx 2^{22,261}$. This special choice of parameters has been used in the original version of Quartz [2], as submitted to NESSIE [10].

4.2 Class of C* Schemes

As HFE, the scheme C^*, due to Matsumoto and Imai [9], uses a finite field $\mathbb{F}$ and an extension field $\mathbb{E}$. However, the choice of the central equation is far more restrictive than in HFE as we only have one monomial here.

Definition 9. *Let $\mathbb{E}$ be an extension field of dimension n over the finite field $\mathbb{F}$ and $\lambda \in \mathbb{N}$ an integer with $\gcd(q^n - 1, q^\lambda + 1) = 1$. We then say that the following central equation is of C^*-shape:*

$$P'(X') := X'^{q^\lambda + 1} .$$

The restriction $\gcd(q^n - 1, q^\lambda + 1) = 1$ is necessary first to obtain a permutation polynomial and second to allow efficient inversion of $P'(X')$. In this setting, we cannot apply the additive sustainer, as this monomial does not allow any linear or constant terms. Moreover, the monomial requires a factor of one. Hence, we have to preserve this property. At present, the only sustainers suitable seem to be the big sustainer (cf Sect. 3.2) and the Frobenius sustainer (cf Sect. 3.6). We use both in the following

Theorem 5. *For $K := (S, P, T) \in \mathit{Aff}(\mathbb{F}^n) \times \mathbb{E}[X] \times \mathit{Aff}(\mathbb{F}^n)$ a private key in C^* we have*

$$n(q^n - 1)$$

equivalent keys. Hence, the key-space of C^* can be reduced by this number.

Proof. To prove this statement, we consider normal forms of keys in C*. In particular, we concentrate on a normal form of the affine transformation S where S is in univariate representation. As for HFE and w.l.o.g., let $B := S(1)$ be a non-zero coefficient on position 1. Unlike to HFE we cannot enforce that $S(0) = 0$, so we may have $S(1) = 0$. However, in this case set $B := S(0)$. Applying $\sigma^{-1}(X) := B^{-1}X$ will ensure a normal form for S. In order to "repair" the monomial $P(X)$, we have to apply an inverse transformation to T. So let $\tau(X) := (B^{q^\lambda+1})^{-1}X$. This way we obtain

$$\begin{aligned}
\mathcal{P} &= T \circ \tau^{-1} \circ \tau \circ P \circ \sigma \circ \sigma^{-1} \circ S \\
&= \tilde{T} \circ (B^{(q^\lambda+1).(-1)}.B^{q^\lambda+1}.X^{q^\lambda+1}) \circ \tilde{S} \\
&= \tilde{T} \circ P \circ \tilde{S} ,
\end{aligned}$$

where $\tilde{S}$ is in normal form. In contrast to HFE (cf Thm. 1), we cannot chose the transformations σ and τ independently: each choice of σ implies a particular τ and vice versa. However, the fix point 1 is still preserved by the Frobenius sustainer and so we can apply this sustainer on the transformation S. As for

HFE, we compute a normal form for a given generator and a total ordering of $\mathbb{E}$; again, we "repair" the monomial $X^{q^\lambda+1}$ by applying an inverse Frobenius sustainer to T and hence have

$$(BX^{q^c}, B^{-q^\lambda-1}X^{q^{n-c}}) \bullet (S,P,T) \text{ where } B \in \mathbb{E}^* \text{ and } 0 \le c < n \text{ for } c \in \mathbb{N}$$

which leads to a total of $n(q^n-1)$ equivalent keys for any given private key. Since all these keys form equivalence classes of equal size, we reduced the private key space of C* by this factor.

Remark 7. Patarin observed that it is possible to derive equivalent keys by changing the monomial P [12]. As the aim of this paper is the study of equivalent keys by chaining the affine transformations S, T alone, we did not make use of this property. A weaker version of the above theorem can be found in [19].

Moreover, we observed in this section that it is not possible for C* to change the transformations S, T from affine to linear. In this context, we want to point out that Geiselmann *et al.* showed how to reveal the constant parts of these transformations [6]. Hence, having S, T affine instead of linear does not seem to enhance the overall security of C*.

For $q = 128$ and $n = 67$, we obtain $\approx 2^{469}$ equivalent private keys per class. The number of choices for S, T is $\approx 2^{63,784}$ in this case.

C^{*--} We want to note that C* itself is insecure, due to a very efficient attack by Patarin [11]. However, for well-chosen parameters q, r, its variation C^{*--} is actually secure: as in the case of HFE and HFE-, we use the original C* scheme and apply the reduction modification from Sect. 3.7.

Theorem 6. *For $K := (S,P,T) \in Aff(\mathbb{F}^n) \times \mathbb{E}[X] \times Aff(\mathbb{F}^n)$ a private key in C* and a reduction number $r \in \mathbb{N}$ we have*

$$n.(q^n-1)q^r \prod_{i=n-r-1}^{n-1} (q^n - q^i)$$

equivalent keys and the key-space of C^{*--} can be reduced by this number.

Proof. This proof is similar to the one of C*, *i.e.*, we apply both the Frobenius and the big sustainer to S and the corresponding inverse sustainer to the transformation T. This way, we "repair" the change on the central monomial $X^{q^\lambda+1}$. All in all, we obtain a factor of $n.(q^n-1)$ equivalent keys for a given private key.

Next we observe that the reduction sustainer applied to the transformation T alone allows us to change the last r rows of the vector $v_T \in \mathbb{F}^n$ and also the last r rows of the matrix $M_T \in \mathbb{F}^{n\times n}$. This yields an additional factor of $q^r \prod_{i=n-r-1}^{n-1}(q^n-q^i)$ on this side.

Note that the changes on the side of the transformation S and the changes on the side of the transformation T actually are independent: the first computes a normal form for S while the second computes a normal form on T. Hence, we may multiply both factors to obtain the overall number of independent keys.

For $q = 128, r = 11$ and $n = 67$, we obtain $\approx 2^{6180}$ equivalent private keys per class. The number of choices for S, T is $\approx 2^{63,784}$ in this case. This particular choice of parameters has been used in Sflash$^{\mathrm{v3}}$ [3].

5 Conclusions

In this paper, we showed through the examples of Hidden Field Equations (HFE) and C* that $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic systems allow many equivalent private keys and hence have a lot of redundancy in this key space, cf Table 1 and Table 2 for numerical examples; the symbols used in Table 1 are explained in the corresponding sections. The $\mathcal{MQ}$-scheme Unbalanced Oil and Vinegar (UOV) has been discussed in [19, Sect. 4.3]. A general overview of $\mathcal{MQ}$-schemes can be found in [20].

Table 1. Summary of the Reduction Results of this Paper

Scheme (*Section*)	Reduction
Hidden Field Equations (*4.1*)	$nq^{2n}(q^n-1)^2$
HFE Minus (*4.1*)	$nq^n(q^n-1)q^{n-r}(q^{n-r}-1)\prod_{i=n-r-1}^{n-1}(q^n-q^i)$
HFE Vinegar (*4.1*)	$n'q^nq^{n'}(q^{n'}-1)^2\prod_{i=0}^{v-1}(q^v-q^i)$
HFE Vinegar Minus (*4.1*)	$n'q^rq^{2n'}(q^{n'}-1)^2\prod_{i=0}^{v-1}(q^v-q^i)\prod_{i=n-r-1}^{n-1}(q^n-q^i)$
C* (*4.2*)	$n(q^n-1)$
C* Minus Minus (*4.2*)	$n(q^n-1)q^r\prod_{i=n-r-1}^{n-1}(q^n-q^i)$

We see applications of our results in different contexts. First, they can be used for memory efficient implementations of the above schemes: using the normal forms outlined in this paper, the memory requirements for the private key can be reduced without jeopardising the security of these schemes. Second, they apply to cryptanalysis as they allow to concentrate on special forms of the private key: an immediate consequence from Sect. 3.1 (additive sustainers) is that HFE does not gain any additional strength from the use of affine rather than linear transformations. Hence, this system should be simplified accordingly. Third, the constructors of new schemes may want to keep these sustaining transformations in mind: there is no point in having a large private key space — if it can be reduced immediately by applying sustainers.

We want to stress that the sustainers from Sect. 3 may not be the only ones possible. We therefore invite other researchers to look for even more powerful transformations. In addition, there are other multivariate schemes which have not been discussed in this paper, due to space and time limitations. These schemes include (non-exhaustive list) enTTS [21], STS [16]), and PMI [4]. We also invite to apply the techniques used in this paper to these schemes to compare the effect of these sustainers to different classes of $\mathcal{MQ}$-schemes.

Table 2. Numerical Examples for the Reduction Results of this Paper

Scheme	Parameters	Choices for S, T (in $\log_2$)	Reduction (in $\log_2$)
HFE	$q = 2, n = 80$	12,056	326
HFE-	$q = 2, r = 7, n = 107$	23,108	2129
HFEv	$q = 2, v = 7, n = 107$	21,652	460
HFEv-	$q = 2, n = 107$	22,261	690
C^*	$q = 128, n = 67$	63,784	469
C^{*--}	$q = 128, n = 67$	63,784	6180

Acknowledgements

We want to thank Patrick Fitzpatrick (BCRI, University College Cork, Ireland) for encouraging this direction of research. In addition, we want to thank An Braeken (COSIC) who pointed out the existence of Frobenius sustainers (cf Sect. 3.6) for fields of even characteristic; in addition we want to thank her for helpful remarks. Moreover, we want to thank Magnus Daum (CTSC, Ruhr-University Bochum, Germany) for comments on some early results presented in this paper.

This work was supported in part by the Concerted Research Action (GOA) GOA Mefisto 2000/06, GOA Ambiorix 2005/11 of the Flemish Government and the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

Disclaimer

The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. An Braeken, Christopher Wolf, and Bart Preneel. A study of the security of Unbalanced Oil and Vinegar signature schemes. In *The Cryptographer's Track at RSA Conference 2005*, Lecture Notes in Computer Science. Alfred J. Menezes, editor, Springer, 2005. 13 pages, cf `http://eprint.iacr.org/2004/222/`.
2. Nicolas Courtois, Louis Goubin, and Jacques Patarin. *Quartz: Primitive specification (second revised version)*, October 2001. `https://www.cosic.esat.kuleuven.ac.be/nessie`Submissions, Quartz, 18 pages.
3. Nicolas Courtois, Louis Goubin, and Jacques Patarin. *SFlash^{v3}, a fast asymmetric signature scheme — Revised Specificatoin of SFlash, version 3.0*, October 17th 2003. ePrint Report 2003/211, `http://eprint.iacr.org/`, 14 pages.

4. Jintai Ding. A new variant of the matsumoto-imai cryptosystem through perturbation. In *Public Key Cryptography — PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 305–318. Feng Bao, Robert H. Deng, and Jianying Zhou (editors), Springer, 2004.
5. Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of Hidden Field Equations (HFE) using gröbner bases. In *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Dan Boneh, editor, Springer, 2003.
6. W. Geiselmann, R. Steinwandt, and Th. Beth. Attacking the affine parts of SFlash. In *Cryptography and Coding - 8th IMA International Conference*, volume 2260 of *Lecture Notes in Computer Science*, pages 355–359. B. Honary, editor, Springer, 2001. Extended version: `http://eprint.iacr.org/2003/220/`.
7. Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar signature schemes. In *Advances in Cryptology — EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Jacques Stern, editor, Springer, 1999.
8. Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem. In *Advances in Cryptology — CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30. Michael Wiener, editor, Springer, 1999. `http://www.minrank.org/hfesubreg.ps` or `http://citeseer.nj.nec.com/kipnis99cryptanalysis.html`
9. Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature verification and message-encryption. In *Advances in Cryptology — EUROCRYPT 1988*, volume 330 of *Lecture Notes in Computer Science*, pages 419–545. Christoph G. Günther, editor, Springer, 1988.
10. NESSIE: New European Schemes for Signatures, Integrity, and Encryption. Information Society Technologies programme of the European commission (IST-1999-12324). `http://www.cryptonessie.org/`.
11. Jacques Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In *Advances in Cryptology — CRYPTO 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 248–261. Don Coppersmith, editor, Springer, 1995.
12. Jacques Patarin. Asymmetric cryptography with a hidden monomial. In *Advances in Cryptology — CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 45–60. Neal Koblitz, editor, Springer, 1996.
13. Jacques Patarin. Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms. In *Advances in Cryptology — EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Ueli Maurer, editor, Springer, 1996. Extended Version: `http://www.minrank.org/hfe.pdf`.
14. Ilia Toli. Cryptanalysis of HFE, June 2003. arXiv preprint server, `http://arxiv.org/abs/cs.CR/0305034`, 7 pages.
15. Christopher Wolf. Efficient public key generation for HFE and variations. In *Cryptographic Algorithms and Their Uses 2004*, pages 78–93. Dawson, Klimm, editors, QUT University, 2004.
16. Christopher Wolf, An Braeken, and Bart Preneel. Efficient cryptanalysis of RSE(2)PKC and RSSE(2)PKC. In *Conference on Security in Communication Networks — SCN 2004*, Lecture Notes in Computer Science, pages 145–151, September 8–10 2004. Extended version: `http://eprint.iacr.org/2004/237`.

17. Christopher Wolf and Bart Preneel. Asymmetric cryptography: Hidden Field Equations. In *European Congress on Computational Methods in Applied Sciences and Engineering 2004*. P. Neittaanmäki, T. Rossi, S. Korotov, E. Oñate, J. Périaux, and D. Knörzer, editors, Jyväskylä University, 2004. 20 pages, extended version: `http://eprint.iacr.org/2004/072/`.
18. Christopher Wolf and Bart Preneel. Equivalent keys in HFE, C*, and variations. In *Proceedings of Mycrypt 2005*, Lecture Notes in Computer Science. Serge Vaudenay, editor, Springer, 2005. Extended version `http://eprint.iacr.org/2004/360/`, 12 pages.
19. Christopher Wolf and Bart Preneel. Superfluous keys in $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic asymmetric systems. In *Public Key Cryptography — PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 275–287. Serge Vaudenay, editor, Springer, 2005. Extended version `http://eprint.iacr.org/2004/361/`.
20. Christopher Wolf and Bart Preneel. Taxonomy of public key schemes based on the problem of multivariate quadratic equations. Cryptology ePrint Archive, Report 2005/077, 12th of May 2005. `http://eprint.iacr.org/2005/077/`, 64 pages.
21. Bo-Yin Yang and Jiun-Ming Chen. Rank attacks and defence in Tame-like multivariate PKC's. Cryptology ePrint Archive, Report 2004/061, 29rd September 2004. `http://eprint.iacr.org/`, 21 pages.

A New Structural Attack for GPT and Variants

Raphael Overbeck

GK Electronic Commerce,
TU-Darmstadt,
Department of Computer Science,
Cryptography and Computer Algebra Group
overbeck@cdc.informatik.tu-darmstadt.de

Abstract. In this paper we look at the Gabidulin version of the McEliece cryptosystem (GPT) and its variants. We propose a new polynomial time attack, which recovers an alternative private key. Our attack is applicable to all variants proposed so far and breaks some of them completely.

Keywords: public key cryptography, code based cryptography, rank distance codes, Gabidulin codes.

1 Introduction

The security of cryptosystems based on error correcting codes is connected to the hardness of the general decoding problem. In 1991 Gabidulin, Paramonov and Tretjakov proposed a variant of the McEliece scheme (GPT) [7] using *rank distance* codes instead of hamming distance codes. Smaller public-key sizes have been proposed for GPT than for the original McEliece cryptosystem, as general decoding algorithms are much slower for the rank metric than for the hamming-metric.

Gibson developed two structural attacks for the GPT cryptosystem (see e.g. [4] and [8]) and proved the parameter sets proposed in [7] and [4] to be insecure. A drawback of Gibson's attacks is, that they have exponential runtime if the secret key is carefully chosen. There were several attempts to modify the GPT cryptosystem, in order to avoid structural attacks, but most of these variants rely on security assumptions very similar to the ones for the original proposal (see [2] and [11]).

In this paper we build a new structural attack on the GPT cryptosystem. Unlike Gibson's attacks it has polynomial runtime, breaks the original GPT cryptosystem from [7] completely and is applicable to all GPT variants proposed so far.

The paper is structured as follows: First we give a short introduction to rank distance codes. Then we present the GPT cryptosystem and its Niederreiter variant. Finally we show how to attack the GPT cryptosystem.

2 Rank Distance Codes

Rank distance codes were presented by Gabidulin in 1985. They are linear codes over the finite field $\mathbb{F}_{q^m}$ for q (a power of a) prime and $m \in \mathbb{N}$. As their name says they use the concept of rank distance.

Definition 1. *Let $x = (x_1, \cdots, x_n) \in \mathbb{F}_{q^m}^n$ and $b_1, \cdots, b_m$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. We can write $x_i = \sum_{j=1}^m x_{ij} b_j$ for each $i = 1, \cdots, n$ with $x_{ij} \in \mathbb{F}_q$. The rank norm $\|x\|_r$ of x is defined as the rank of the matrix $(x_{ij}) \in \mathbb{F}_{q^m}^{n \times m}$.*

The rank norm of a vector $x \in \mathbb{F}_{q^m}^n$ is uniquely determined (independent of the choice of basis) and induces a metric, called *rank distance.*

Definition 2. *An (n, k)-code $\mathcal{G}$ over a finite field $\mathbb{F}$ is a k-dimensional subvectorspace of the vector space $\mathbb{F}^n$. We call the code $\mathcal{G}$ an (n, k, d) rank distance code if $d = \min_{x,y \in \mathcal{G}} \|x - y\|_r$. The matrix $G \in \mathbb{F}^{k \times n}$ is a generator matrix for the (n, k) code $\mathcal{G}$ over $\mathbb{F}$, if the rows of G span $\mathcal{G}$ over $\mathbb{F}$. The matrix $H \in \mathbb{F}^{n \times (n-k)}$ is called check matrix for the code $\mathcal{G}$ if it is the right kernel of G. The code generated by $H^\top$ is called dual code of $\mathcal{G}$ and denoted by $\mathcal{G}^\perp$.*

In [9] Ourivski and Johansson presented an algorithm which solves the general decoding problem in $\mathcal{O}\left((m\frac{d-1}{2})^3 q^{(d-3)(k+1)/2}\right)$ operations over $\mathbb{F}_q$ for (n, k, d) rank distance codes over $\mathbb{F}_{q^m}$. A special class of rank distance codes are the *Gabidulin codes* for which an efficient decoding algorithm exists [4]. We will define these codes by their generator matrix.

Definition 3. *Let $k \le n \le m \in \mathbb{N}$ and $g \in \mathbb{F}_{q^m}^n$ be a vector s.t. the components g_i, $i = 1, \cdots, n$ are linearly independent over $\mathbb{F}_q$. The (n, k, d) Gabidulin code $\mathcal{G}$ is the rank distance code with generator matrix*

$$G = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^q & g_2^q & \cdots & g_n^q \\ \vdots & & \ddots & \vdots \\ g_1^{q^{k-1}} & g_2^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{pmatrix} \in \mathbb{F}_{q^m}^{k \times n}. \tag{1}$$

An (n, k) Gabidulin code $\mathcal{G}$ corrects $\left\lfloor \frac{n-k}{2} \right\rfloor$ errors and has a minimum distance of $d = n - k + 1$. The dual code of an (n, k) Gabidulin code is an $(n, n - k)$ Gabidulin code (see [4]). The vector g is said to be a *generator vector* of the Gabidulin code $\mathcal{G}$ (it is not unique). Error correction based on the *right Euclidean division algorithm* takes $\mathcal{O}\left(d \log_2^2 d + dn\right)$ operations over $\mathbb{F}_{q^m}$ for (n, k, d) Gabidulin codes [4].

Throughout this paper we will use the following notation. We write $\mathcal{G} = \langle G \rangle$ if the (n, k)-code $\mathcal{G}$ over the field $\mathbb{F}$ has the generator matrix G. If the rows of a $(n - k) \times n$ matrix M span $\mathcal{G}^\perp$ we write $G^\perp = M$. We will identify $x \in \mathbb{F}^n$ with $(x_1, \cdots, x_n), x_i \in \mathbb{F}$ for $i = 1, \cdots, n$. For any (ordered) subset $\{j_1, \cdots j_m\} =: J \subseteq \{1, \cdots n\}$ we denote the vector $(x_{j_1}, \cdots, x_{j_m}) \in \mathbb{F}^m$ with x_J. Similarly, for a $k \times n$ matrix M we denote by $M_{\cdot J}$ the submatrix consisting of the columns corresponding to the indices of J and write $M_{J'\cdot} = \left(\left(M^\top\right)_{\cdot J'}\right)^\top$ for any (ordered) subset J' of $\{1, \cdots, k\}$. Block matrices will be given in brackets.

3 The GPT Cryptosystem

The GPT cryptosystem was first presented in 1991 by Gabidulin, Paramonov and Tretjakov [7]. Here we present a more generalized version (GGPT, see [11]), which may be used to describe the original GPT cryptosystem as well as the variant with column scrambler from [3].

- **System Parameters:** $q, k < n \leq m$, $s \leq t \in \mathbb{N}$, where $t < n - k - 1$.
- **Key Generation:** First generate the following matrices :
  $G \in \mathbb{F}_{q^m}^{k \times n}$ generator matrix of an (n, k, d) Gabidulin code,
  $X \in \mathbb{F}_{q^m}^{k \times t}$ random matrix of rank s over $\mathbb{F}_{q^m}$ and rank t over $\mathbb{F}_q$,
  $S \in \mathbb{F}_{q^m}^{k \times k}$ random, non-singular matrix (the row scrambler) and
  $T \in \mathbb{F}_q^{n \times n}$ random, non-singular matrix (the column scrambler).

  Then compute the $k \times n$ matrix

  $$\begin{aligned} G' &= S\left(\left[\, X \middle| 0 \,\right] + G\right) T \\ &= S\left[\, G_{\cdot\{1,\cdots,t\}} + X \middle| G_{\cdot\{t+1,\cdots,n\}} \,\right] T \in \mathbb{F}_{q^m}^{k \times n} \ , \end{aligned} \tag{2}$$

  where 0 denotes the $k \times (n-t)$ zero matrix. Choose $1 \leq e \leq \frac{n-k-t}{2}$. Further let $\mathcal{D}_{\mathcal{G}}$ be an efficient decoding algorithm for the Gabidulin code $\mathcal{G}$ generated by the matrix $G_{\cdot\{t+1,\cdots,n\}}$.
- **Public Key:** (G', e)
- **Private Key:** $(\mathcal{D}_{\mathcal{G}}, S, T)$ or (G, S, T) where G is of the form in (1).
- **Encryption:** To encode a plaintext $x \in \mathbb{F}_{q^m}^k$ choose a vector $z \in \mathbb{F}_{q^m}^n$ of rank norm e at random and compute the ciphertext c as follows:

  $$c = xG' + z \ .$$

- **Decryption:** To decode a ciphertext c apply the decoding algorithm $\mathcal{D}_{\mathcal{G}}$ for $\mathcal{G}$ to $c' = \left(cT^{-1}\right)_{\{t+1,\cdots,n\}}$. As T is a invertible matrix over $\mathbb{F}_q$, the rank norm of a vector does not change if it is multiplied with T^{-1}. Thus c' has at most rank distance $\frac{n-k-t}{2}$ to $\mathcal{G}$ and we obtain the codeword

  $$xSG_{\{t+1,\cdots,n\}} = \mathcal{D}_{\mathcal{G}}\left(c'\right) \ .$$

  Now, we can compute the plaintext x.

In the original GPT cryptosystem, the parameterse e and t are chosen such that $e = \frac{n-k}{2} - t$. If we do so, the legitimate user may recover $xSGT$ by applying the error correction algorithm for $\langle GT \rangle$ (which is a Gabidulin code, too) to the ciphertext c.

The distortion matrix X is essential to mask the structure of G. We can recover the vector gT from SGT in $\mathcal{O}\left(k^3\right)$ operations over $\mathbb{F}_{q^m}$ by employing methods similar to the attack of Sidelnikov and Shestakov on the Niederreiter cryptosystem using GRS codes (see [4]). If the parameter s should be larger than $t/2$, as there exists a polynomial time attack on the private key [8]. In all examples we will choose $n = m$ and $q = 2$. Some parameter sets may be found in table 3 (All of these are secure against all previously published attacks).

3.1 The Niederreiter Variant of GPT

The security of the GGPT cryptosystem is strongly connected to the one of the Niederreiter variant, as we will see later on. We briefly introduce the Niederreiter variant of the GPT cryptosystem from [2]. On key generation we choose a $k-l$ dimensional subcode of an (n,k) Gabidulin code $\mathcal{G}$ over $\mathbb{F}_{q^m}$. Every check matrix of the subcode may be described as

$$(H') = \left[H\middle|A\right] S \in \mathbb{F}_{q^m}^{n\times(n-k+l)},$$

where H is the $n \times (n-k)$ check matrix of $\mathcal{G}$, A is an $n \times l$ matrix of full rank and S is some invertible $(n-k+l)\times(n-k+l)$ matrix. The public key $(H', e = (n-k)/2)$ is published, and the pair $(S, \mathcal{G})$ is taken to be the private key. To encode a plaintext $x \in \mathbb{F}_{q^m}^n$ of rank norm less then e, compute the ciphertext c as follows:

$$c = xH' \ .$$

In order to decode a ciphertext c apply the syndrome decoding algorithm $\mathcal{D}_{\mathcal{G}}$ for $\mathcal{G}$ to the syndrome build from the first $n-k$ columns of cS^{-1}. Table 1 shows public key sizes and approximate work factors (WF = operations over $\mathbb{F}_q$) for the fastest general decoding attack. Parameters were taken from [1].

Table 1. Parameter sets for the Niederreiter GPT

Parameters			Size Public	WF general
m	k	l	Key (Bytes)	decoding
25	15	5	469	2^{82}
32	24	4	960	2^{93}

4 Attacking the GPT Cryptosystem

Even though there were attempts to break the GPT cryptosystem by using general rank distance decoding algorithms, the structural attacks from Gibson (see e.g. [4], [8]) had more impact on the cryptosystem. However, for carefully chosen parameter sets, Gibson's attacks have exponential running time (see appendix). Several variants of GPT were proposed, but it was shown, that the security of the variants from [3] and [6] is connected to the security of GGPT (see [11]). The attempt to use Gibson's attack to cryptanalyze these variants failed for the variant from [6], but resulted in an attack for the variant from [3].

The main weakness of the GPT cryptosystem is, that it is difficult to hide the structure of the generator matrix of a Gabidulin code. As already noted by Gibson, the use of subfield subcodes (or group codes) seems much more promising for cryptographic applications. Here, we want to use some observations on Gabidulin codes: For a matrix M let $M^{[j]}$ denote the result of rising every element of M to the power of j. If G is the generator matrix of a Gabidulin code, then G and $G^{[q]}$ look quite the same. (Both define Gabidulin codes with generator vectors g and $g^{[q]}$ respectively.) We are going to use this property to distinguish the Gabidulin part of the public code from the random one.

Let M be an arbitrary $l \times n$ matrix over $\mathbb{F}_{q^m}$ and $f \in \mathbb{N}$. While Gibson analyzed matrices of the form $M + M^{[q]}$ (compare [8]), we look at matrices of the form

$$\Lambda_f(M) := \begin{bmatrix} M \\ (M)^{[q]} \\ \vdots \\ (M)^{[q^f]} \end{bmatrix} \in \mathbb{F}_{q^m}^{((f+1)\cdot l)\times n}. \tag{3}$$

Lemma 1. *If $M \in \mathbb{F}_{q^m}^{l\times n}$ defines an (n,k) Gabidulin code with generator vector g and $f \leq n-k-1$, then the subvectorspace spanned by the rows of $\Lambda_f(M)$ defines the $(n,k+f)$ Gabidulin code with generator vector g.*

Assumption 1. *Let $M \in \mathbb{F}_{q^m}^{l\times n}$ define a random $l > 1$ dimensional subcode of an (n,k) Gabidulin code over $\mathbb{F}_{q^m}$ with generator vector g. Then with probability $\mathcal{P}_1 \geq (1-q^{-m})$, $\Lambda_f(M)$ defines a* $\min\{k+f,(f+1)\cdot l\}$ *dimensional subcode of the $(n,k+f)$ Gabidulin code with generator vector g.*

Assumption 2. *Let $M \in \mathbb{F}_{q^m}^{l\times n}$ be a random matrix of full rank over $\mathbb{F}_{q^m}$ and of full column rank over $\mathbb{F}_q$. Then $\Lambda_f(M)$ has rank* $\min(n, f\cdot l)$ *with probability $\mathcal{P}_2 \geq (1-q^{-(m-1)})$.*

The proof for lemma 1 is obvious. For assumption 1, it is easy to see, that $\Lambda_f(M)$ defines a subcode of the $(n,k+f)$ Gabidulin code with generator vector g, so the remaining part is to estimate $\mathcal{P}_1$. Assumption 2 is based on empirical results as well as on observations from [5]. If $l = 1$, then because of theorem 1, the assumption is true, as $\mathcal{P}_2 = 1$. Experiments for parameters relevant for our attacks showed that $\mathcal{P}_1$ and $\mathcal{P}_2$ are almost 1 (see appendix). However, not the correctness, but only the success probability of the attacks proposed in the following sections depends on the assumptions above.

4.1 Attacking the Niederreiter Variant

The Niederreiter variant of the GPT cryptosystem was first attacked by A. Ourivski in [10]. Here we present a new attack, which recovers an alternative secret key in polynomial time by using assumption 1.

Theorem 1. *Let $\mathcal{G}_{\mathrm{SUB}}$ be a random $k-l$ dimensional subcode of an (n,k) Gabidulin code $\mathcal{G}$ over $\mathbb{F}_{q^m}$ with generator vector g. Then we may recover g from $\mathcal{G}_{\mathrm{SUB}}$ with probability $\mathcal{P}_1$ if $k-l > 1$ and $n-k-1 \geq \lceil l/(k-l-1)\rceil$. Further, this may be done in $\mathcal{O}(n^3)$ operations over $\mathbb{F}_{q^m}$.*

Proof. Let G' be the generator matrix of $\mathcal{G}_{\mathrm{SUB}}$. To recover g from $\mathcal{G}_{\mathrm{SUB}}$ we choose $f \in \mathbb{N}$ such that $n-k-1 \geq f \geq \lceil l/(k-l-1)\rceil$. If assumption 1 holds, $\Lambda_f(G')$ has rank $k+f$ with probability $\mathcal{P}_1$ and defines a subcode of a $(n,k+f)$ Gabidulin code. Thus, with probability $\mathcal{P}_1$, $\Lambda_f(G')$ spans the $(n,k+f)$ Gabidulin code with generator vector g and we can recover g in $\mathcal{O}((k+f)^3)$ operations over $\mathbb{F}_{q^m}$ (see [4]).

It follows, that if assumption 1 holds, we can recover the secret Gabidulin code $\mathcal{G}$ from the public key of an instance of the Niederreiter variant of GPT as long as $n-k-1 \geq \lceil l/(k-l-1) \rceil$. Let H be the check matrix of $\mathcal{G}$. To obtain an equivalent secret key, we can choose a set J of l columns of H', s.t. the matrix $\left[H \middle| H'_{\cdot J} \right]$ has full rank. Now we may solve the equation

$$H' = \left[H \middle| H'_{\cdot J} \right] \bar{S}$$

for $\bar{S}$ and obtain the alternative secret key $\left(\mathcal{G}, \bar{S}\right)$. Note, that employing this method, it only takes $\mathcal{O}\left((k+f)^3\right)$ operations over $\mathbb{F}_{q^m}$ to recover an alternative secret key.

For the parameter sets proposed e.g. in [1], the choice of $f = 1$ showed to be sufficient in all our experiments. Table 2 shows modified parameter sets for which the presented attack does not work. These parameters are not necessarily secure (see [10]).

Table 2. Modified parameter sets for the Niederreiter GPT

Parameters			Public Key	WF general
m	k	l	Size (Bytes)	decoding
32	24	20	448	2^{93}
64	52	47	2360	2^{288}

4.2 Attacking the GPT Cryptosystem

To recover an alternative secret key from the public key (G', e) of an instance of the GGPT cryptosystem, we want to use assumption 2. The general idea is, to observe the behavior of the matrix $\Lambda_f(G')$. We assume, that if the difference of the rank of $\Lambda_f(G')$ and $\Lambda_{f+1}(G')$ is only 1 for some f, then $\Lambda_f(G')$ will be strongly connected to a Gabidulin code. The following theorem describes the connection:

Theorem 2. *Let (G', e) be the public key of an instance of the GGPT cryptosystem with parameters q, m, n, k, t and s. Further, let (G, S, T) be the corresponding secret key. Then for $0 \leq f \leq n-t-k-1$, there exists a dual matrix of $\Lambda_f(G')$ of the form*

$$\Lambda_f(G')^{\perp} = \begin{bmatrix} 0 & H_f^{\top} \\ B_1 & B_2 \end{bmatrix} \cdot \left(T^{-1}\right)^{\top} \in \mathbb{F}_{q^m}^{(n-t-k-f+l)\times n}, \tag{4}$$

where $H_f \in \mathbb{F}_{q^m}^{(n-t)\times(n-t-k-f)}$ is the check matrix of a $k+f$ dimensional Gabidulin code $\mathcal{G}_f$ of length $n-t$, B_1 is some $l \times t$ matrix with $0 \leq l \leq t$ and B_2 is some $l \times (n-t)$ matrix.

Proof. First, we assume, that T and S are the identity matrix. The proof is analogous, if this is not the case. We may write

$$\varLambda_f\left(G'\right) = \left[\,\varLambda_f\left(G_{\cdot\{1,\cdots,t\}} + X\right)\middle|\varLambda_f\left(G_{\cdot\{t+1,\cdots,n\}}\right)\right] \in \mathbb{F}_{q^m}^{(kf)\times n}$$

By assumption 2, the last $n-t$ columns of $\varLambda_f\left(G'\right)$ define an $(n-t, k+f)$ Gabidulin code $\mathcal{G}_f$. Thus the subvectorspace spanned by the rows of

$$\left[\,0\middle|H_f^\top\,\right] \in \mathbb{F}_{q^m}^{(n-t-k-f)\times n},$$

where $H_f \in \mathbb{F}_{q^m}^{(n-t)\times(n-t-k-f)}$ is the check matrix of $\mathcal{G}_f$, is in the dual space of $\varLambda_f\left(G'\right)$. To get a matrix which defines the whole dual space of $\varLambda_f\left(G'\right)$, we might have to add some more rows to $\left[\,0\middle|H_f^\top\,\right]$. However, it is clear, that there will be at most t rows missing, as $\varLambda_f\left(G'\right)$ has at least rank $k+f$. This proves the theorem.

Observe, that $\mathcal{G}_f$ is uniquely defined by the secret key and f. Thus, knowing H_f for some f, we know all H_i for $0 \leq i \leq n-k-t-1$. We are going to determine the rank of $\varLambda_f\left(G'\right)^\perp$ in the following sections. For now, we assume, that it will be very near its lower bound $(n-t-k-f)$ and show, how to recover an alternative secret key in that case (compare example in the appendix).

Theorem 3. *Let (G', e) be as in theorem 2. Given an $f \leq n-t-k-1$ s.t. the rank of $\varLambda_f\left(G'\right)^\perp$ is $n-t-k-f$, then we may recover an alternative secret key, corresponding to G' in $\mathcal{O}\left(n^3\right)$ operations over $\mathbb{F}_{q^m}$.*

Proof. With the conditions above, it follows from theorem 2, that there is a matrix $\varLambda_f\left(G'\right)^\perp$ of the form

$$\left[\,0\middle|H_f^\top\,\right]\left(T^{-1}\right)^\top \in \mathbb{F}_{q^m}^{(n-t-k-f)\times n},$$

where H_f is as in theorem 2. We can recover such a matrix in $\mathcal{O}\left(n^3\right)$ operations over $\mathbb{F}_{q^m}$ [4]. Now we can choose a set N_1 of $n-t$ rows of G' s.t. $\varLambda_f\left(G'\right)^\perp_{\cdot N_1}$ is of column rank $n-t$ over $\mathbb{F}_q$. It follows, that $T_{N_1 N_2}$ with $N_2 = \{t+1, \cdots, n\}$ is invertible. We may assume without loss of generality that $N_1 = N_2$ and $H_f^\top = \varLambda_f\left(G'\right)^\perp_{\cdot N_1}$. Let $\tilde{T} \in \mathbb{F}_q^{t\times(n-t)}$ be the solution of the equation

$$\varLambda_f\left(G'\right)^\perp_{\cdot\{1,\cdots,t\}} = H_f^\top \cdot \tilde{T}^\top$$

over $\mathbb{F}_q$. We define

$$\bar{T}^{-1} := \begin{bmatrix} \mathrm{Id}_t & \tilde{T} \\ 0 & \mathrm{Id}_{n-t} \end{bmatrix} \in \mathbb{F}_q^{n\times n},$$

where Id_k denotes the k-dimensional identity matrix. We may recover H_0 from H_f, as both are uniquely determined by G' and $\bar{T}$. It follows, that $\left[\,0\middle|H_0\,\right]$ is in the dual space of $G'\bar{T}^{-1}$, and thus the last $n-t$ columns of $G'\bar{T}^{-1}$ define an $(n-t, k)$ Gabidulin code. Thus $\bar{T}$ serves as an alternative column scrambler. Now, we may obtain an equivalent secret key in $\mathcal{O}\left(k^3\right)$ operations by applying the methods from [4] to $\left(G' \cdot \bar{T}^{-1}\right)_{\cdot\{t+1,\cdots,n\}}$, which gives us an alternative row scrambler $\bar{S}$.

However, even if the rank of $\varLambda_f\left(G'\right)$ is larger than $n-t-k-f$, an attacker still may try to recover the secret key. He could guess a set N_1 of $n-t$ rowss.t. ($\left[\,0\middle|H_f\right]\left(T^{-1}\right)^\top$)$_{\cdot N_1}$ has full column rank over $\mathbb{F}_q$. Again we may assume w.l.o.g. that $N_1 = N_2$ and $\left(T^{-1}\right)_{N_1 N_2} = \mathrm{Id}_{n-t}$. Then, the matrix $\left(\Lambda_f\left(G'\right)^{\perp}_{\cdot N_1} \right)^{\top}$ corresponds to an instance of the Niederreiter version of GPT as long as $k+f-l>1$. Thus, we might apply the attacks on the Niederreiter variant of GPT, to recover $\left[H_f \middle| B_2^{\top}\right]$. If one of the attacks succeeds, an attacker can recover a dual matrix of $\Lambda_f\left(G'\right)$ of the form given in equation (4) and from it an alternative column scrambler. Afterwards the attacker would be able to construct a valid alternative private key.

4.3 Strengths of the New Attack

Given an f s.t. the conditions of theorem 3 are fulfilled, for the GGPT public key, we can build an alternative private key in $\mathcal{O}\left(m^5\right)$ operations over $\mathbb{F}_q$. By now, we have no idea, for which parameter sets our attack might work. To estimate the success probability of our attack, we will have to determine the size of $\Lambda_f\left(G'\right)^{\perp}$. In the following we assume that $s<k$.

Theorem 4. *Let (G', e) be as in theorem 2. If assumption 2 holds for $s \times t$ matrices over $\mathbb{F}_{q^m}$, then the rank of the dual matrix of $\Lambda_f\left(G'\right)$ is at most $R = n-k-f-\min\{t, fs\}$ with probability $\mathcal{P}_2$.*

Proof. **(Theorem 4)** We have to estimate the rank of $\Lambda_f\left(G'\right)$ for given $G' = S\left(\left[X\middle|0\right]+G\right)T$ and f (see equation 2). To simplify notations, we define the following matrices:

$$M_k := \begin{bmatrix} 0 & \mathrm{Id}_{(k-1)} \\ 0 & 0 \end{bmatrix} \in \mathbb{F}_{q^m}^{k\times k},\ \gamma_i := \left(\left[X\middle|0\right]+G\right)_{k\cdot}^{\left[q^i\right]} \in \mathbb{F}_{q^m}^{1\times n} \text{ and}$$

$$\tilde{X}_i := \left(X^{\left[q^{i-1}\right]}\right)_{\cdot\{2,\cdots,k\}} + \left(X^{\left[q^i\right]}\right)_{\cdot\{1,\cdots,k-1\}} \in \mathbb{F}_{q^m}^{k-1\times t}\ .$$

To determine the rank of $\Lambda_f\left(G'\right)$ we use the property: If G is of the form in equation (1), then the result of adding the $(j+1)$-th row of $G^{\left[q^i\right]}$ to the j-th row of $G^{\left[q^{i+1}\right]}$ is zero for $0 \le i \le f-1$ and $1 \le j \le k-1$. Thus, by removing the influence of S from $\Lambda_f\left(G'\right)$ and adding the rows as mentioned above by using M_k, we get the following matrix of the same rank as $\Lambda_f\left(G'\right)$:

$$\begin{bmatrix} \mathrm{Id}_k & 0 & \cdots & 0 \\ M_k & \mathrm{Id}_k & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & M_k & \mathrm{Id}_k \end{bmatrix} \cdot \begin{bmatrix} S^{\left[q^0\right]} & 0 & \cdots & 0 \\ 0 & S^{\left[q^1\right]} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & S^{\left[q^f\right]} \end{bmatrix}^{-1} \cdot \Lambda_f\left(G'\right)$$

$$= \left[\begin{array}{cc} X + G_{\cdot\{1,\cdots,t\}} & G_{\cdot\{t+1,\cdots,n\}} \\ \hline \tilde{X}_1 & 0 \\ \hline \multicolumn{2}{c}{\gamma_1} \\ \hline \vdots & \vdots \\ \hline \tilde{X}_f & 0 \\ \hline \multicolumn{2}{c}{\gamma_f} \end{array}\right] \cdot T \in \mathbb{F}_{q^m}^{((f+1)\cdot k)\times n}\ .$$

With probability $\mathcal{P}_2$ the part of the matrix above build from the $[\tilde{X}_i\ 0]$ contains at least $\min\{t, fs\}$ linearly independent rows, as $(\tilde{X}_i)^{[q]} = \tilde{X}_{i+1}$ and $\operatorname{rank}(X_i) = s$. Therefore $\Lambda_f(G')$ has at least rank $k + f + \min\{t, fs\}$ with probability $\mathcal{P}_2$.

Note, that for $s = 1$ assumption 2 is correct, and the probability in the theorem above gets 1. Otherwise the conditions in theorem 3 are fulfilled with probability $\mathcal{P}_2$. We conclude, that all parameter sets, where there exists an $f \leq n - k - t - 1$, s.t. $t \leq fs$ are insecure. Furthermore, as $s \geq 1$, we may obtain a equivalent secret key from the public key with probability 1 for all parameter sets where

$$t \leq n - k - t - 1 \iff 1/2 \leq (n-k)/2 - t \ , \tag{5}$$

even if $s > 1$. This is true for all instances of the original GPT cryptosystem.

4.4 Experimental Results

Table 3 shows absolute run times the attack by methods from theorem 3 in comparison to the theoretical work factors (operations over $\mathbb{F}_q$) of the previous attacks. For all parameter sets we chose $f = n - t - k - 1$. In our experiments our attack did not fail for any random instance of the original GPT cryptosystem. Operations were performed on a 500MHz Pentium III running Linux using an implementation in Java.

Table 3. Attacking the GPT cryptosystem

Parameters				average runtime	WF best of	WF general
m	k	t	s	of our attack	Gibson's attacks	decoding
48	10	16	3	51 min	2^{139}	2^{134}
48	16	18	4	58 min	2^{200}	2^{124}
48	24	8	2	102 min	2^{122}	2^{198}

In our experiments we chose X as the product of a random $k \times s$ matrix S_X of rank $s < k$ over $\mathbb{F}_{q^m}$ and a random $s \times t$ matrix $\bar{X}$ (of rank s over $\mathbb{F}_{q^m}$ and rank t over $\mathbb{F}_q$). For such choices of X the matrix $\Lambda_f(G')$ almost always had rank $(k + f + (s+1) \cdot \min(f, s) + s \cdot \max(0, f - s))$ or $k + f + t$. For special choices of S_X and random $\bar{X}$, we were able to create instances, where the rank of $\Lambda_f(G')$ reached the bound R. However, choosing S_X or $\bar{X}$ of a special form removes degrees of freedom in choosing the private key and thus does not seem to be a good choice.

4.5 On Secure Instances of GGPT

We have seen, that instances of the GPT cryptosystem and its variants, where

$$t \leq s \cdot (n - t - k - 1)$$

holds, are insecure if assumption 2 holds. For the GGPT variant however, we may choose parameter sets, s.t. this equation does not hold. Even though, we might be able recover an equivalent private key if we can choose an f s.t. $k+f-t+fs > 1$, as described in section 4.2.

To get secure instances of the GGPT cryptosystem, one could try to choose parameters in a way, such that $t-fs > f+k$ for every possible choice of f. The latter is the case, e.g. if

$$s \leq \frac{2t-n}{n-t-k} \ .$$

A parameter set satisfying this condition would be $n = m = 64$, $k = 8$, $t = 40$ and $s = 1$ e.g. with a public key size of 3584 bytes. The attack in the given form is not applicable for such parameter sets. However, it seems very likely that the attack may be modified in such a way, that these parameter sets can be attacked, too.

5 Conclusion

We conclude that the original GPT cryptosystem from [7] is broken by our attack. Our attacks succeed with good probability for most parameter sets of GGPT and can even be extended to other variants of the GPT cryptosystem (compare [6], [11] and [3]). After several attacks on the GPT cryptosystem and its variants, it seems to be difficult to name secure parameter sets for GGPT, if there exist any. Even if we would consider the parameter set mentioned above to be secure, the GPT cryptosystem looses much of its advantages over the McEliece cryptosystem.

References

1. T.P. Berger and P. Loidreau. How to mask the structure of codes for a cryptographic use. *Designs, Codes and Cryptography*, 35 (1), 2005.
2. T.P. Berger and P. Loidreau. Security of the Niederreiter form of the GPT public-key cryptosystem. In *IEEE International Symposium on Information Theory, Lausanne, Suisse*. IEEE, July 2002.
3. E. M. Gabidulin and A. V. Ourivski. Column scrambler for the GPT cryptosystem. *Discrete Applied Mathematics*, 128(1):207–221, 2003.
4. E.M. Gabidulin. On public-key cryptosystems based on linear codes. In *Proc. of 4th IMA Conference on Cryptography and Coding 1993*, Codes and Ciphers. IMA Press, 1995.
5. E.M. Gabidulin and P. Loidreau. Subfield subcodes of maximum-rank distance codes. In *Seventh International Workshop on Algebraic and Combinatorial Coding Theory*, volume 7 of *ACCT*, pages 151–156, 2000.
6. E.M. Gabidulin, A.V. Ourivski, B. Honary, and B. Ammar. Reducible rank codes and their applications to cryptography. *IEEE Transactions on Information Theory*, 49(12):3289–3293, 2003.
7. E.M. Gabidulin, A.V. Paramonov, and O.V. Tretjakov. Ideals over a non-commutative ring and their applications to cryptography. In *Proc. Eurocrypt '91*, volume 547 of *LNCS*. Springer Verlag, 1991.

8. K. Gibson. The security of the Gabidulin public key cryptosystem. In *Proc. of Eurocrypt'96*, volume 1070 of *LNCS*, pages 212–223. Springer Verlag, 1996.
9. T. Johansson and A.V. Ourivski. New technique for decoding codes in the rank metric and its cryptography applications. *Problems of Information Transmission*, 38, No. 3:237–246, 2002.
10. A.V. Ourivski. Recovering a parent code for subcodes of maximal rank distance codes. In *Proc. of WCC 03*, pages 357–363, 2003.
11. R. Overbeck. Extending Gibson's attacks on the GPT cryptosystem. In *Proc. of WCC 2005*, pages 382–391, 2005.

A Appendix - Gibson's Attacks

Gibson presented two structural attacks on the GPT cryptosystem. They recover an alternative private-key from the GGPT public-key G'. On input of $G' = S\left(\left[X|0\right] + G\right)T$, Gibson's attacks return $\hat{G}$, $\hat{X} \in \mathbb{F}_{q^m}^{k\times n}$ and $\hat{S} \in \mathbb{F}_{q^m}^{k\times k}$, s.t.

(i) $\hat{G}$ is a generator matrix of an (n,k) Gabidulin code over $\mathbb{F}_{q^m}$,
(ii) $G' = \hat{S}\left(\hat{G} + \hat{X}\right)$ and
(iii) the rank of $\hat{X}$ over $\mathbb{F}_q$ is not bigger than t.

Thus Gibson's attacks serve well for an attack on the GGPT cryptosystem, as an alternative column scrambler may be recovered from $\hat{X}$. Gibson's first attack was developed for the case that the GGPT parameter s is 1, but may be adapted to the case where $s \neq 1$ (see [4]). It takes

$$\mathcal{O}\left(m^3 (n-k)^3 q^{ms}\right) \tag{6}$$

operations over $\mathbb{F}_{q^m}$. In [8] Gibson presented a different attack, which is more efficient for larger values of s. It requires that $k + t + 2 \leq n$ and runs in

$$\mathcal{O}\left(k^3 + (k+t)\, f \cdot q^{f(k+2)} + (m-k)\, t \cdot q^f\right) \tag{7}$$

operations over $\mathbb{F}_{q^m}$, where $f \approx \max(0, t-2s, t+1-k)$. Note, that this attack runs in polynomial time if $f = 0$. The success of both attacks is based on some assumptions, which are claimed to be fulfilled with high probability for random instances of the GGPT cryptosystem. Nevertheless Gibson's attacks are not fast enough to attack the GGPT cryptosystem for all parameter sets of practical interest (compare Table 3).

B Appendix - On Assumption 1

Besides our experimental results, we want to give some intuition, why assumption 1 seems to be reasonable. Let $G' = \bar{S}G$, where $\bar{S} \in \mathbb{F}_{q^m}^{(k-l)\times k}$ is of full rank and G is the generator matrix of the (n,k) Gabidulin code with generator vector g.

Now G' defines a subcode of the code generated by G. Let $\bar{G}$ be the generator matrix of the $(n, k+f)$ Gabidulin code with generator vector g. We may write

$$(G')^{[q^i]} = [\ \underbrace{\mathbf{0}|\cdots|\mathbf{0}}_{i \text{ times}}\ |\bar{S}^{[q^i]}|\ \underbrace{\mathbf{0}|\cdots|\mathbf{0}}_{f-i \text{ times}}\ ]\,\bar{G} \in \mathbb{F}_{q^m}^{(k-l)\times n}\ ,$$

where $\mathbf{0}$ is the $k \times 1$ matrix with only zero entries. Then $\Lambda_f\,(G')$ may be written as

$$\Lambda_f\,(G') = \underbrace{\begin{bmatrix} \bar{S} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \bar{S}^{[q]} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \bar{S}^{[q^f]} \end{bmatrix}}_{=:S} \bar{G} \in \mathbb{F}_{q^m}^{(f+1)(k-l)\times n}.$$

If $\bar{S}$ is a random matrix, then S seems to be of full rank, with high probability. In our experiments we did not find any counterexamples for randomly generated matrices $\bar{S}$, where we chose (n, k) Gabidulin codes with $n \geq 8$, $k \geq 4$ and the dimension of the subcode to be $l \geq 2$.

C Appendix - On Assumption 2

In order to estimate the probability $\mathcal{P}_2$ in assumption 2 we made several experiments for random matrices M. In all our experiments, we build $\Lambda_f\,(M)$ for all $1 \leq f \leq \lceil m/l \rceil$, where m is the extension degree of the field, and l is the rank of the matrix M. Table 4 shows the resulting probability estimates. We conclude, that it is reasonable to assume, that $1 - \mathcal{P}_2$ decreases exponentially fast with growing m.

Table 4. Experimental results for assumption 2

rows	columns	field	$\mathcal{P}_2$ estimate	# experiments
2	6	$\mathbb{F}_{q^6}$	$1 - 0.0289$	10000
2	6	$\mathbb{F}_{q^8}$	$1 - 0.0050$	10000
2	6	$\mathbb{F}_{q^{10}}$	$1 - 0.0010$	10000
2	8	$\mathbb{F}_{q^{10}}$	$1 - 0.0008$	10000
2	10	$\mathbb{F}_{q^{10}}$	$1 - 0.0018$	10000
3	10	$\mathbb{F}_{q^{10}}$	$1 - 0.0$	10000
4	10	$\mathbb{F}_{q^{10}}$	$1 - 0.0$	10000
5	10	$\mathbb{F}_{q^{10}}$	$1 - 0.0013$	10000
2	8	$\mathbb{F}_{q^{16}}$	$1 - 0.000033$	30000
2	10	$\mathbb{F}_{q^{16}}$	$1 - 0.0$	30000

D Appendix - A Small Example

For a better understanding of the attack on the GGPT cryptosystem presented in the previous sections, we provide an example with small parameters: $q = 2$, $m = n = 5$, $k = 2$, $t = s = 1$. As field we choose $\mathbb{F}_{q^5} = \mathbb{F}_2 / \left(X^5 + X^2 + 1\right)$. We write the elements of this field in their polynomial representation, thus $X^3 + 1 \mathrel{\hat{=}} 01001$.

Assume, that we are given a public key (G', e) with $e = 1$ and

$$G' = \begin{pmatrix} 10101\ 10011\ 00111\ 01011\ 01111 \\ 00010\ 11001\ 10000\ 10011\ 00011 \end{pmatrix} .$$

The (unknown) secret key is (G, S, T) with

$$S = \begin{pmatrix} 10000\ 10100 \\ 01000\ 10000 \end{pmatrix} \text{ and } T = \begin{pmatrix} 1\ 0\ 1\ 1\ 1 \\ 0\ 1\ 0\ 0\ 0 \\ 1\ 1\ 0\ 1\ 1 \\ 0\ 1\ 1\ 1\ 0 \\ 1\ 0\ 0\ 0\ 0 \end{pmatrix} .$$

To recover an alternative secret key, an attacker would choose the parameter $f = n - k - t - 1 = 1$ and build

$$\Lambda_f(G') = \begin{pmatrix} 10101\ 10011\ 00111\ 01011\ 01111 \\ 00010\ 11001\ 10000\ 10011\ 00011 \\ 11100\ 01000\ 10101\ 01111\ 11111 \\ 00100\ 00110\ 01101\ 01000\ 00101 \end{pmatrix} .$$

The dual of $\langle \Lambda_f(G') \rangle$ is defined by

$$\Lambda_f(G')^{\perp} = \begin{pmatrix} 00100\ 01110\ 01100\ 01001\ 00001 \end{pmatrix} .$$

The attacker observes, that the last 4 columns of $\Lambda_f(G')^{\perp}$ are linearly independent over $\mathbb{F}_2$, which is the rank of $\Lambda_f(G')^{\perp}$ over $\mathbb{F}_2$. The legitimate user would be able to compute

$$\Lambda_f(G')^{\perp} T^{\top} = \left[0 \middle| H_1 \right] = \begin{pmatrix} 00000\ 01110\ 00010\ 01011\ 00100 \end{pmatrix} .$$

The attacker on the other hand can choose

$$\left(\Lambda_f(G')^{\perp} \right)_{\{2,\cdots,5\}} = \begin{pmatrix} 01110\ 01100\ 01001\ 00001 \end{pmatrix}$$

to be his H_1. (He could choose any other submatrix of column rank 4 over $\mathbb{F}_2$, and each would lead to a different alternative secret key.) As a solution to the equation $(00100) = H_1 \tilde{T}^{\top}$ the attacker gets $\tilde{T} = \begin{pmatrix} 0\ 1\ 1\ 1 \end{pmatrix}$. Now,

$$G' \cdot \begin{bmatrix} 1 & \tilde{T} \\ 0 & \mathrm{Id}_4 \end{bmatrix}^{-1} = \begin{pmatrix} 10101\ 10011\ 10010\ 11110\ 11010 \\ 00010\ 11001\ 10010\ 10001\ 00001 \end{pmatrix} .$$

The last four columns of the matrix above define a Gabidulin code with generator vector $\begin{pmatrix} 01010\ 01001\ 00100\ 00001 \end{pmatrix}$. Thus, the attacker gets the row scrambler

$$\bar{S} = \begin{pmatrix} 11001\ 00011 \\ 11110\ 11111 \end{pmatrix}$$

and obtains a working alternative secret key.

A Family of Fast Syndrome Based Cryptographic Hash Functions

Daniel Augot[1], Matthieu Finiasz[1,2], and Nicolas Sendrier[1]

[1] Projet Codes, INRIA Rocquencourt,
BP 105, 78153 Le Chesnay - Cedex, France
{Daniel.Augot,Matthieu.Finiasz,Nicolas.Sendrier}@inria.fr
[2] LASEC, École Polytechnique Fédérale de Lausanne (EPFL),
Station 14, 1015 Lausanne, Switzerland

Abstract. Recently, some collisions have been exposed for a variety of cryptographic hash functions [20,21] including some of the most widely used today. Many other hash functions using similar constructions can however still be considered secure. Nevertheless, this has drawn attention on the need for new hash function designs.

In this article is presented a family of secure hash functions, whose security is directly related to the syndrome decoding problem from the theory of error-correcting codes.

Taking into account the analysis by Coron and Joux [4] based on Wagner's generalized birthday algorithm [19] we study the asymptotical security of our functions. We demonstrate that this attack is always exponential in terms of the length of the hash value.

We also study the work-factor of this attack, along with other attacks from coding theory, for non asymptotic range, i.e. for practical values. Accordingly, we propose a few sets of parameters giving a good security and either a faster hashing or a shorter description for the function.

Keywords: cryptographic hash functions, provable security, syndrome decoding, NP-completeness, Wagner's generalized birthday problem.

1 Introduction

The main cryptographic hash function design in use today iterates a so called compression function according to Merkle's and Damgård's constructions [6,12]. Classical compression functions are very fast [13,16] but, in general, cannot be proven secure. However, provable security may be achieved with compression functions designed following public key principles, at the cost of being less efficient. This has been done for instance by Damgård in [6], where he designed a hash function based on the Knapsack problem. Accordingly, this function has been broken by Granboulan and Joux [8], using lattice reduction algorithms. The present paper contributes to the hash function family by designing functions based on the syndrome decoding problem, which is immune to lattice reduction based attacks.

Unlike most other public key cryptosystems, the encryption function of the McEliece cryptosystem [10] (or of Niederreiter's version [14]) is nearly as fast as a symmetric cipher. Using this function with a random matrix instead of the usual parity check matrix of a Goppa code, a provably secure one-way function has been constructed in [1]: since there is no trapdoor, its security can be readily related to the difficulty of syndrome decoding. For instance, there is no polynomial time algorithm to decode a random code, thus there is no polynomial time algorithm to invert the compression function and/or find a collision.

However, for the practical parameters which have been proposed in [1], there is an efficient attack with a cost as low as 2^{43} (or 2^{62} depending on the set of parameters), as demonstrated by Coron and Joux [4], using Wagner's method for the generalized birthday problem [19].

The purpose of this paper is to propose updated parameters for the hash function family presented in [1]. We do not only extend the parameters to be out of reach of the Coron-Joux attack, but we also thoroughly study the asymptotical behavior of their attack. We shall establish that this attack is exponential, such that the design for the hash function is sound.

The paper is organized as follows. In Section 2, we introduce the *Fast Syndrome Based* (FSB) compression function, derived from a hard problem similar to syndrome decoding. In Section 3 we show that the security of FSB is reduced to the average case difficulty of two new NP-complete problems. Then, in Section 4, we show how the best known decoding techniques, and the new method based on Wagner's ideas, can be adapted to the cryptanalysis of our functions. From that we can evaluate the practical security and the scalability of the system. In Section 5, we propose some choices of parameters and, eventually, we compare the obtained functions with other existing constructions. For clarity of the presentation, NP-completeness proofs are postponed in the appendix.

2 The Hash Function

We present what is called the *Fast Syndrome Based* (FSB) hash function in [1].

2.1 General Construction of Hash Functions

We follow Merkle's and Damgård's design principle of hash functions [6,12]: iterating a compression function (here denoted $\mathcal{F}$), which takes as input s bits and returns r bits (with $s > r$). The resulting function is then chained to operate on strings of arbitrary length (see Fig. 1). The validity of such a design has been established [6,12], and its security is proven not worse than the security of the compression function. Therefore we will only concentrate on the security of the latter.

2.2 Description of the Compression Function

The core of the compression function is a random binary matrix $\mathcal{H}$ of size $r \times n$. The parameters for the hash function are:

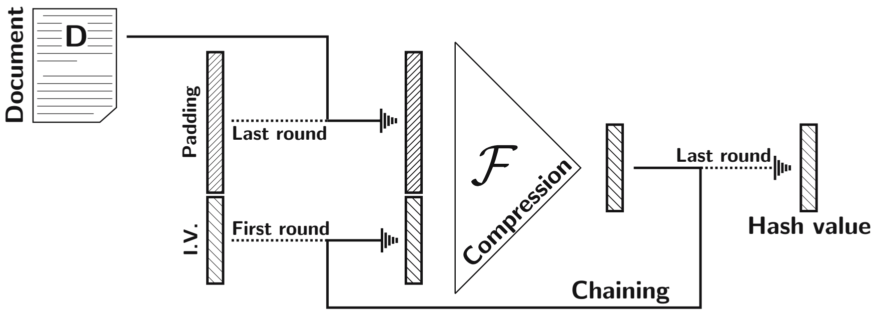


Fig. 1. A standard hash function construction

- n the number of columns of $\mathcal{H}$;
- r the number of rows of $\mathcal{H}$ and the size in bits of the function output;
- w the number of columns of $\mathcal{H}$ added at each round.

Definition 1. *A word of length n and weight w is called* regular *if it has exactly one non-zero position in each of the w intervals $\left[\!\left[(i-1)\frac{n}{w}; i\frac{n}{w}\right]\!\right]_{i=1..w}$. We call a* block *such an interval.*

In order to encode a regular word of length n and Hamming weight w, $s = w\log_2(\frac{n}{w})$ bits are needed. This is the size in bits of the input of the compression function $\mathcal{F}$. When practical parameters will be chosen, it will be made in such a manner that round figures for $\log_2(\frac{n}{w})$ are obtained. That is $\frac{n}{w}$ has to be a power of 2 and ideally, for software efficiency, $\log_2(\frac{n}{w})$ too.

The matrix $\mathcal{H}$ is split into w sub-blocks $\mathcal{H}_i$, of size $r \times \frac{n}{w}$, and the algorithm describing $\mathcal{F}$ is:

```
                                        FSB compression function
Input: $s$ bits of data
    1. split the $s$ input bits in $w$ parts $s_1, \dots, s_w$ of $\log_2(\frac{n}{w})$ bits;
    2. convert each $s_i$ to an integer between 1 and $\frac{n}{w}$;
    3. choose the corresponding column in each $\mathcal{H}_i$;
    4. add the $w$ chosen columns to obtain a binary string of length $r$.
Output: $r$ bits of hash
```

The speed of $\mathcal{F}$ is directly related to the number of XORs required at each round: one needs to XOR w columns of r bits, that is wr binary XORs. The number of bits read in the document at each round is $w\log_2\frac{n}{w} - r$ (input size minus chaining size). Hence, the average number of binary XORs required for each document input bit is:

$$\mathcal{N}_{XOR} = \frac{r \cdot w}{w\log_2\frac{n}{w} - r}.$$

This value will be the right measure for the global speed of the FSB hash function.

3 Theoretical Security

As stated in [11,17], a cryptographic hash function has to be pre-image resistant, second pre-image resistant and collision resistant. As the second pre-image resistance is strictly weaker than collision resistance, we will only check that the hash function is collision free and resistant to inversion. We show that the inversion and collision finding are related to two problems very close to syndrome decoding, which is a hard problem [3]. We describe them here and show (in appendix) that they are also NP-complete.

3.1 Two New NP-Complete Problems

Regular Syndrome Decoding (RSD)
Input: w matrices $\mathcal{H}_i$ of dimension $r \times \frac{n}{w}$ and a bit string $\mathcal{S}$ of length r.
Property: there exists a set of w columns, one in each $\mathcal{H}_i$, summing to $\mathcal{S}$.

Definition 2. *A* 2-regular word *is a word of weight less than or equal to* $2w$*, which contains either* 0 *or* 2 *non zero positions in each block. It is the sum of two regular words.*

2-Regular Null Syndrome Decoding (2-RNSD)
Input: w matrices $\mathcal{H}_i$ of dimension $r \times \frac{n}{w}$.
Property: there exists a set of $2w'$ columns (with $0 < w' \leq w$), 0 or 2 in each $\mathcal{H}_i$, summing to 0.

Thus solving **2-Regular Null Syndrome Decoding** requires to find a non null 2-regular word of weight less than or equal to $2w$.

3.2 Security Reduction

In this section we will recall how finding collisions or inverting the FSB hash function is exactly as hard as solving an instance of one of the NP-complete problems described in the previous section.

Let us be given an algorithm for the *inversion* of the compression function, which, given a hash value $\mathcal{S}$, finds an inverse m such that $\mathcal{F}(m) = \mathcal{S}$, in time T with probability p of success. Then it is a tautology that this algorithm is able to solve any instance of the problem **Regular Syndrome Decoding**, in the same time and with the same probability of success. Similarly an algorithm which is able to find a collision gives in fact two different regular words c_1 and c_2 of weight w such $Hc_1^t = Hc_2^t$. Then $c = c_1 \oplus c_2$ is a non null 2-regular word and has a null syndrome. So c is directly a solution for **2-Regular Null Syndrome Decoding**.

These reductions to NP-complete problems only prove worst case difficulty. However, following Gurevich and Levin [7,9] discussion for syndrome decoding, we believe that both these problems are difficult on average.

3.3 Average Case Consideration

From a cryptographic point of view, knowing that some instances of a problem are hard is not enough to consider it a hard problem. It is more important that the number of *weak instances* is small enough, that is, the probability of having to solve such a weak instance when attacking the system is negligible.

However, defining a weak instance is not so simple as it will depend on the algorithm used to attack the system: the instances solved with the smallest complexity will vary when changing algorithm. A weak instance should hence be defined as an instance which is weak for at least one algorithm: an instance for which one algorithm yields a noticeably smaller complexity than the average complexity of the best algorithm.

A problem should not be considered hard if the proportion of those weak instances among the total number of possible instances is not negligible. When trying to find collisions for FSB, each binary $r \times n$ matrix defines a different instance. In Section 4.6, after seeing the possible attacks on the system, we will try to estimate the proportion of these matrices defining such a weak instance.

4 Practical Security

We recall the possible practical attacks on the compression function $\mathcal{F}$, and study the minimal work-factors required to perform these attacks. There are two kinds of algorithms: Information Set Decoding and Wagner's Generalized Birthday Paradox. We will survey the results on Information Set Decoding algorithm, which has been studied in [1]. As for Wagner's Generalized Birthday Paradox [19], we will give an extended analysis: first we slightly generalize Wagner's algorithm, then we describe how its complexity is exponential when the length of the hash value goes to infinity.

4.1 Information Set Decoding

The problem of decoding a random code has been extensively studied and many algorithms devoted to this task have been developed (see [2] for a survey). All these algorithms are exponential. Still, as stated by Sendrier [18], the most efficient attacks all seem to be derived from *Information Set Decoding* (ISD).

Definition 3. *An* information set *is a set of $k = n - r$ (the dimension of the code) positions among the n positions of the support.*

Definition 4. *Let $((\mathcal{H}_i)_{1\leq i\leq w}, w, \mathcal{S})$ be an instance of RSD. An information set will be called* valid *with respect to the instance if there exists a solution to this problem which has no 1s among the k positions of the set.*

The ISD algorithm consists in picking information sets at random, until a valid one is found. Checking whether the information set is valid or not is done in polynomial time[1], so the exponential nature of the algorithm originates from the exponentially small probability of finding a valid information set: let $V(r)$ be the cost of checking the validity of an information set, and $\mathcal{P}_w$ the probability

[1] one simply has to perform a Gaussian elimination on the matrix, using the columns outside the chosen information set.

for a random information set to be valid; then the complexity of this algorithm is $V(r)/\mathcal{P}_w$.

The probability $\mathcal{P}_w$ depends on the probability $\mathcal{P}_{w,1}$, that a given information set is valid for one given solution of RSD, and on the expected number $\mathcal{N}_w$ of solutions to RSD. We shall consider:

$$\mathcal{P}_w = 1 - (1 - \mathcal{P}_{w,1})^{\mathcal{N}_w}.$$

For simplicity, we will use the convenient approximation $\mathcal{P}_w \simeq \mathcal{P}_{w,1} \times \mathcal{N}_w$.

In the case of RSD, one needs to find a *regular* word of weight w having a given syndrome $\mathcal{S}$. The number of regular solutions to RSD is, on average:

$$\mathcal{N}_w = \frac{\left(\frac{n}{w}\right)^w}{2^r}.$$

As the solutions are not random words, the attacker should first choose the information sets which have the best chance of being valid. One can see that he will maximize his chances when choosing the same number of positions in each block, that is, choosing $\frac{k}{w}$ positions w times. The probability of success is then:

$$\mathcal{P}_{w,1} = \left(\frac{\binom{n/w-1}{k/w}}{\binom{n/w}{k/w}} \right)^w = \frac{\left(\frac{r}{w}\right)^w}{\left(\frac{n}{w}\right)^w}.$$

The final probability $\mathcal{P}_{\text{inv}}$ of choosing a valid set to invert RSD is: $\mathcal{P}_{\text{inv}} = \mathcal{P}_{w,1} \times \mathcal{N}_w = \frac{\left(\frac{r}{w}\right)^w}{2^r}$. Note that it does not depend on n.

For *collisions*, one needs to find a 2-regular word with null syndrome. If i is the number of non-zero blocks in this word, the number of such words is:

$$\mathcal{N}_i = \frac{\binom{w}{i}\binom{n/w}{2}^i}{2^r}.$$

Using, as for inversion, an equal repartition of the information set among the blocks, that is, $\frac{k}{w}$ positions in each block, we get, for each value of i, the probability of validity:

$$\mathcal{P}_{i,1} = \frac{\binom{w}{i}\binom{n/w-k/w}{2}^i}{\binom{w}{i}\binom{n/w}{2}^i} = \frac{\binom{r/w}{2}^i}{\binom{n/w}{2}^i}.$$

The total probability of success for one information set is then:

$$\mathcal{P}_{\text{col total}} = \sum_{i=1}^{w} \frac{\binom{w}{i}\binom{r/w}{2}^i}{2^r} = \frac{1}{2^r}\left[\binom{\frac{r}{w}}{2} + 1\right]^w.$$

However the adversary may decide to use another strategy and look for specific words. He can consider words with non-zero positions only in a given set of $w_0 < w$ blocks, take all the information set points available in the remaining

$w - w_0$ blocks and distribute the rest of the information set in the w_0 chosen blocks. The probability of success is then:

$$\mathcal{P}_{\text{col } w_0} = \frac{1}{2^r}\left[\binom{\frac{n}{w} - \frac{k_0}{w_0}}{2} + 1\right]^{w_0} = \frac{1}{2^r}\left[\binom{\frac{r}{w_0}}{2} + 1\right]^{w_0},$$

with $k_0 = k - (w - w_0) \times \frac{n}{w} = \frac{n \cdot w_0}{w} - r$. As the attacker has the possibility to choose the best strategy, he can choose the most suitable value for w_0 (as long as it remains smaller than w):

$$\mathcal{P}_{\text{col optimal}} = \frac{1}{2^r} \max_{w_0 \in [\![1;w]\!]} \left[\binom{\frac{r}{w_0}}{2} + 1\right]^{w_0}.$$

It is shown in [1] that this maximum is reached for $w_0 = \alpha \cdot r$, where $\alpha \approx 0.24$ is a constant, and that:

$$\mathcal{P}_{\text{col optimal}} = \frac{1}{2^r}\left[\binom{\frac{1}{\alpha}}{2} + 1\right]^{\alpha r} \simeq 2^{\frac{r}{3.3}}.$$

4.2 Wagner's Generalized Birthday Problem

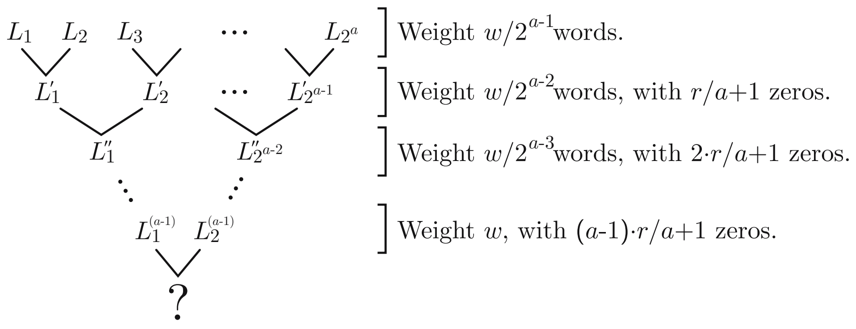
We now describe the attack from Coron and Joux [4], which relies on the Generalized Birthday Problem introduced by Wagner [19], who established:

Theorem 1. *Let L_1, L_2,...,L_{2^a} be 2^a lists of r bits strings, each list being of size $L = 2^{\frac{r}{a+1}}$. Then a solution to the equation $x_1 \oplus x_2 \oplus \cdots \oplus x_{2^a} = 0$, with $x_i \in L_i$, can be found in time $O(2^a 2^{\frac{r}{a+1}})$.*

Let us recall the algorithm. At first step 2^a lists of size L are given. Then the lists are merged pair by pair to create a new list: for instance, the merge $L_1 \bowtie L_2$ of the lists L_1 and L_2 is made from the sum of the elements x_1, x_2 of L_1, L_2 such that $x_1 \oplus x_2$ is zero on the first $\frac{r}{a+1}$ bits. One readily checks that the size $L_1 \bowtie L_2$ is still $2^{\frac{r}{a+1}}$ on average. Using sorting algorithms, this merge operation can be done in time $O(r 2^{\frac{r}{a+1}})$ (one can eliminate the r factor, and thus obtain Wagner's complexity, using hash tables, however this will require larger memory space). Once the 2^{a-1} new lists are obtained, one proceeds recursively, until there are only two remaining lists (See Fig. 2). Then a collision is found between these two lists using the classical birthday paradox. Since there are 2^a merge operations, the total complexity is $O(r 2^a 2^{\frac{r}{a+1}})$.

This algorithm translates into an attack for collisions as follows. Each list L_i is associated to $\frac{w}{2^a}$ blocks of the matrix, and contains the syndromes of 2-regular words, of weight less than or equal to $\frac{2w}{2^a}$, defined over these blocks. Finding a solution $Hx_1 \oplus Hx_2 \oplus \cdots \oplus Hx_{2^a} = 0$ gives a collision.

The adversary will try to optimize a in order to get the lowest complexity. But the auxiliary a is subject to the following constraint: each list (of size $L = 2^{\frac{r}{a+1}}$) can not be larger than the number of words of weight $\frac{2w}{2^a}$ (or lower), which are



L_1 L_2 L_3 ⋯ L_{2^a}
Weight $w/2^{a\text{-}1}$words.
L'_1 L'_2 ⋯ $L'_{2^{a-1}}$
Weight $w/2^{a\text{-}2}$words, with $r/a{+}1$ zeros.
L''_1 $L''_{2^{a-2}}$
Weight $w/2^{a\text{-}3}$words, with $2{\cdot}r/a{+}1$ zeros.
$L_1^{(a\text{-}1)}$ $L_2^{(a\text{-}1)}$
Weight w, with $(a\text{-}1){\cdot}r/a{+}1$ zeros.
?

Fig. 2. Application of Wagner's algorithm to collision search. All the lists remain of constant size $L = 2^{\frac{r}{a+1}}$. On average, there remains a single solution at the end of the algorithm.

part of a 2-regular word, with a given set of blocks. Since in each block there are $\binom{\frac{n}{w}}{2} + 1$ words of weight 2 or 0, this gives:

$$\frac{r}{a+1} \leq \frac{w}{2^a} \log_2 \left[\binom{\frac{n}{w}}{2} + 1 \right],$$

or equivalently:

$$\frac{2^a}{a+1} \leq \frac{w}{r} \log_2 \left[\binom{\frac{n}{w}}{2} + 1 \right]. \tag{1}$$

This shows that a can not grow as desired by the adversary. In the case of inversion search, the constraint is that the size of the list must be smaller than the number of regular words of weight $\frac{w}{2^a}$, with 1's in some $\frac{w}{2^a}$ blocks. This gives, similarly to the collision case:

$$\frac{2^a}{a+1} \leq \frac{w}{r} \log_2 (\frac{n}{w}). \tag{2}$$

4.3 Extension of Wagner's Algorithm to Non-fitting Cases

In general, it may be the case that the size $L = 2^\ell$ of the lists to be dealt with is not exactly $2^{\frac{r}{a+1}}$.

We first deal with the case when $\ell < \frac{r}{a+1}$. In that case, we apply Wagner's method, with the constraint of zeroing ℓ bits of the partial sums (instead of $\frac{r}{a+1}$) during each merge operation, hence keeping lists of constant size. So, the two remaining lists, at the end of the recursion, will only have $(a-1)\ell$ bits equal to zero. Then the probability to have a collision between these two lists is $\frac{2^{(a+1)\ell}}{2^r}$. If the algorithm fails and there is no collision, then the whole process is started from the beginning, choosing another set of bits to be set to zero. Since the complexity of building the two final lists is $O(\ell 2^a 2^\ell)$, the total cost is $O(\ell 2^{r+a-a\ell})$. Again, for this complexity to hold, the size of the list must be smaller than $2^{\frac{r}{a+1}}$. The contrary would correspond to having more than one collision in the end, which won't help improving the complexity.

Secondly, we deal with the case when $\ell > \frac{r}{a+1}$. Here the strategy is to prepare each list using a precomputation step, by zeroing a few bits in a each list. We shall denote by α this number of bits and calculate its optimal value. After zeroing α bits, the size of the lists is on average $2^{\ell-\alpha}$. In the context of the hash function, this precomputation step is performed by using two sublists and merging them using the birthday paradox. Then Wagner's algorithm is applied to set to zero the $r' = r - \alpha$ remaining bits. Ideally α is chosen to fit the new parameters of Wagner's algorithm: we must have $\ell - \alpha = \frac{r'}{a+1}$. Solving these two equations gives:

$$\alpha = \frac{\ell(a+1) - r}{a} \text{ and } r' = \frac{a+1}{a}(r - \ell),$$

and the total cost of Wagner's algorithm is $O(r' 2^a 2^{\frac{r'}{a+1}})$. Note that preparing all the lists, with the birthday paradox, costs $O(2^a 2^{\frac{\ell}{2}})$, so there might be a concern that this step becomes preponderant. Solving the inequalities tells us that this is only the case when $\ell > \frac{2r}{a+2}$, which means that we fall in the range where $a+1$ could have been used for the attack (see Equations (1) and (2)).

4.4 Some Security Curves

The curves on Fig. 3 show how the different attacks described in this section behave when applied to concrete parameters. It is clear that the attack based on Wagner's Generalized Birthday Paradox gives far better results than Information Set Decoding techniques applied to regular words. This is mainly because Information Set Decoding algorithms are efficient when applied to a problem having a single solution. This is often the case when decoding a random code, but here, each instance has a huge number of solutions.

It is also important to note that, for a same security level, the scope of available parameters is much wider if one is only looking for a one-way function (no collision resistance). For instance, with $r = 400$ and $n = 2^{16}$, for a security of 2^{80}operations, w could be chosen anywhere in the interval $[\![0;145]\!]$ instead of $[\![0;67]\!]$.

4.5 Asymptotical Behavior

We want to prove that, even though a can vary depending on the parameters, when r goes to infinity a can not grow fast enough to make Wagner's attack sub-exponential. The only constraint on a is:

$$\frac{2^a}{a+1} \leq \frac{w}{r} \log_2\left(\frac{n}{w}\right).$$

If we consider n and w as polynomial in r (noted $\mathcal{P}oly\,(r)$), then $\frac{n}{w}$ is also polynomial in r and we have:

$$\frac{2^a}{a+1} \leq \mathcal{P}oly\,(r) \log_2\left(\mathcal{P}oly\,(r)\right).$$

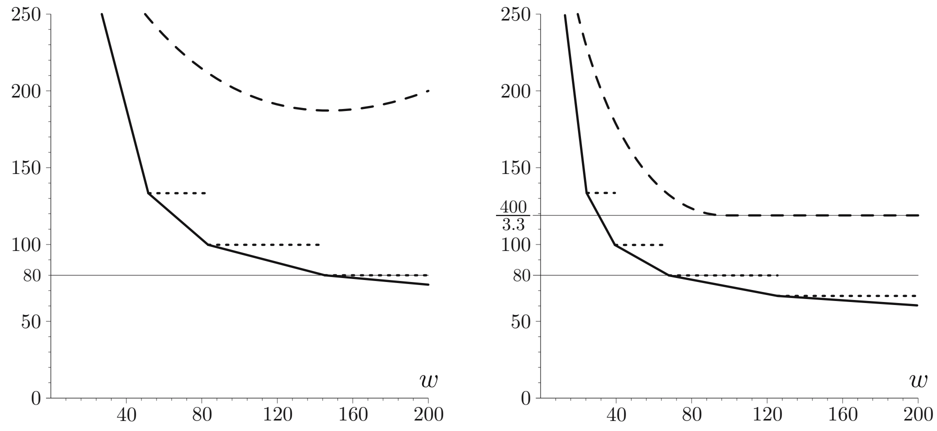


Fig. 3. Comparison of the costs of the different attacks as a function of w when $r = 400$ and $n = 2^{16}$. On the left when applied to inversion and on the right to collision search. The vertical scale corresponds to the logarithm of the work-factor required to perform Information Set Decoding (*dashed line*), Wagner's Generalized Birthday Paradox (*dotted line*) or Extended Wagner Paradox (*plain line*).

From this we deduce $a = \mathcal{P}oly\,(\log_2 r)$. Asymptotically, the best attack having a cost of $O(r 2^a 2^{\frac{r}{a+1}})$ remains thus exponential in r. Moreover, it should be possible to find parameters which scale well when trying to increase the security level.

The simplest solution to achieve so is to scale the parameters linearly with r. For instance, suppose we have two constants ω and ν such that $w = \omega \times r$ and $n = \nu \times r$. We get:

$$\frac{2^a}{a+1} \leq \omega \log_2 \left(\frac{\nu}{\omega}\right) \quad \text{so} \quad a \simeq \log_2 \omega \log_2 \log_2 \frac{\nu}{\omega} = \kappa \quad \text{a constant.}$$

This means that the best a an attacker can use will remain constant whatever the value of r. Asymptotically this construction will scale with:

- exponential security: $2^{\frac{r}{a+1}} = 2^{\mathcal{O}(r)}$,
- constant block size: $\log_2 \frac{n}{w} = \log_2 \frac{\nu}{\omega} = $ constant ($\frac{n}{w}$ remains a power of 2),
- linear hash cost: $\mathcal{N}_{XOR} = \frac{r^2\omega}{r(\omega \log \frac{\nu}{\omega} - 1)} = \mathcal{O}\left(r\right)$,
- quadratic matrix size: $r \times n = r^2\nu = \mathcal{O}\left(r^2\right)$.

Using this method it is possible to convert any set of efficient parameters to another efficient set giving any other required security, by simply scaling r.

4.6 Weak Instances

As defined in Section 3.3, a weak instance will correspond to a matrix $\mathcal{H}$ for which there exists an algorithm being able to find a collision (a 2-regular word having a null syndrome) with a complexity lower than that of the best attack: here the

Wagner-based attack. We shall hence go over known attacks and evaluate the number of weak instances each one will generate.

Instances for which the Wagner-based attack can have a complexity lower than the average are those for which, when using smaller lists all along the algorithm, there remains, on average, one solution in the end. However, this is only possible if the matrix is not of full rank: the algorithm can then be applied using $r' = \text{Rank}(\mathcal{H})$ instead of r. However, the probability of $\mathcal{H}$ not being full rank is very small (about $\mathcal{O}\left(2^{r-n}\right)$) and these weak instances can be neglected.

Concerning the Information Set Decoding attack, instances which will have a low complexity will also represent a negligible proportion of all possible matrices: it requires that there is an abnormally large amount of solutions of low weight, so that when choosing a random information set, it will have a larger probability of being valid. This probability will be even smaller than that of $\mathcal{H}$ not being of full rank.

The only remaining property which could weaken an instance against collision search is the presence of a very low weight collision (say $2w_0$). This way, a brute force search among all low weight words could find this collision with a lower complexity. A search among all words of weight up to $2w_0$ will have, using the birthday paradox, a complexity of $\mathcal{O}\left(\sqrt{\mathcal{N}_{w_0}}\right)$ where $\mathcal{N}_{w_0}$ denotes the number of 2-regular words of weight $2w_0$. This attack will hence only be of interest for values of w_0 such that $\mathcal{N}_{w_0} < 2^{\frac{2r}{a+1}}$ (the square of the average complexity of the Wagner-based attack). The probability that an instance has a solution of weight $2w_0$ is $\mathcal{O}\left(\frac{\mathcal{N}_{w_0}}{2^r}\right)$ and thus the proportion of such weak instances can not be larger than $\mathcal{O}\left(2^{(a-1)\frac{-r}{a+1}}\right)$, which, as we will see in the next section, will always be negligible for secure parameters.

We can hence conclude that no known attack yields a non negligible proportion of weak instances in our construction.

5 Proposed Parameters

Usually hash functions have a security of $2^{\frac{r}{2}}$ against collisions, that is, the best attack is based on the classical birthday paradox. In our case this would correspond to a being equal to 1 at most. However, if this is the case, $\mathcal{F}$ will necessarily have an input shorter than its output and this is incompatible with the chaining method. If we want to have an input size larger than the output size (i.e. compression), then the attacker will always have the possibility to choose at least $a = 3$ (when looking for collisions). If we suppose our parameters place us exactly at a point where an attacker can use $a = 3$, we have:

$$r = \frac{(3+1)w}{2^3}\log_2\left[\binom{\frac{n}{w}}{2}+1\right] \geq \frac{w}{2}\log_2\left[\left(\frac{n}{w}\right)^2 \times \frac{1}{2}\right] = w\log_2\left(\frac{n}{w}\right) - \frac{w}{2}.$$

If this is the case we will then have:

$$\mathcal{N}_{XOR} \geq \frac{rw}{\frac{w}{2}} = 2r.$$

For a security of 2^{80} and with $a = 3$ we would need at least $r = 320$ and hence at least 640 binary XORs per input document bit. This is not so huge in practice but it would still give a relatively slow hash rate. For instance, it is just above 10 Mbits/s, using a vanilla C implementation on a Pentium 4.

If we instead choose to limit the attacker to $a = 4$ we will have much more freedom in the parameter choice. First of all, we get:

$$\mathcal{N}_{XOR} = \frac{rw}{\left(1 - \frac{5}{8}\right)\log_2\left(\frac{n}{w}\right) + \frac{5}{16}w}.$$

Changing the values of n and w (which are still linked by the constraint $a = 4$) will let us change the hash cost. However, as we see on Fig. 4, the lowest values for $\mathcal{N}_{XOR}$ also correspond to the largest values of n and so, to larger matrix sizes. Table 1 collects a list of parameter sets all corresponding to $a = 4$ and $r = 400$, that is, a security of 2^{80}. In fact, practical security will be a little higher as we have neglected a $r2^a$ factor. The security should hence rather be around 2^{92} operations, and an exact security of 2^{80} would be achieved with $r = 340$. However an attacker with huge memory can get rid of the r factor. We will hence stick to the $2^{\frac{r}{a+1}}$ approximation of the security.

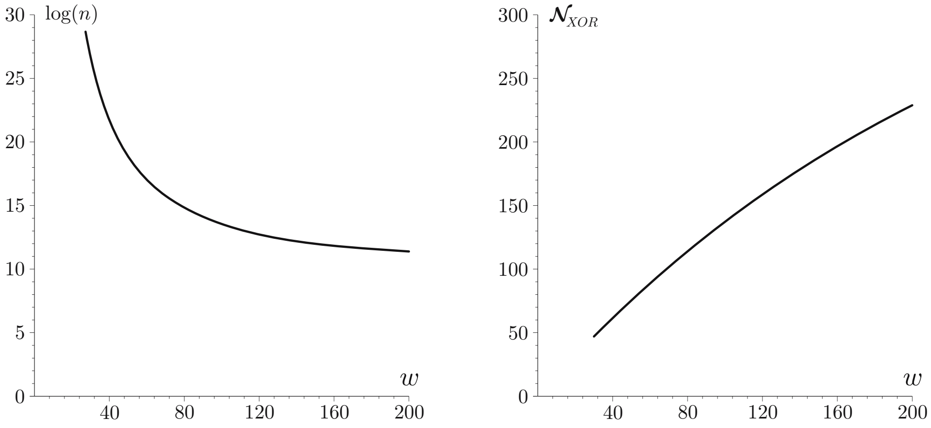


Fig. 4. Evolution of $\log_2 n$ (*on the left*) and $\mathcal{N}_{XOR}$ (*on the right*) as a function of w for $r = 400$ when always staying as close as possible to the point $a = 4$

If we choose to use the set of parameters where $\log_2 \frac{n}{w} = 8$, which is very convenient for software implementation, we will have at the same time a good efficiency (7 times faster than when trying to force $a = 3$) and a reasonable matrix size. As we have seen in Section 4.5, we can then scale these parameters. We have $\omega = \frac{85}{400} = 0.2125$ and $\nu = 256 \times \omega = 54.4$. If we now want to hash with a security of 2^{128} (equivalent to that of SHA-256) we simply need to use $r = 128 \times (a + 1) = 640$ and so $w = 640 \times \omega = 136$ and $n = 34816$.

Table 1. Possible parameters for $r = 400$ and $a = 4$

$\log_2\left(\frac{n}{w}\right)$	w	n	$\mathcal{N}_{XOR}$	matrix size
16	41	2 686 976	64.0	$\sim$ 1 Gbit
15	44	1 441 792	67.7	550 Mbits
14	47	770 048	72.9	293 Mbits
13	51	417 792	77.6	159 Mbits
12	55	225 280	84.6	86 Mbits
11	60	122 880	92.3	47 Mbits
10	67	68 608	99.3	26 Mbits
9	75	38 400	109.1	15 Mbits
8	85	21 760	121.4	8.3 Mbits
7	98	12 544	137.1	4.8 Mbits
6	116	7 424	156.8	2.8 Mbits
5	142	4 544	183.2	1.7 Mbits
4	185	2 960	217.6	1.1 Mbits

We propose three sets of parameters giving a security of 2^{80} against collision search for different output hash sizes. Each of these sets can be scaled linearly to obtain a better security.

- **Short Hash:** $r = 320$, $w = 42$ and $\log_2 \frac{n}{w} = 8$. This solution has a hash size of 320 bits only, but is quite slow with a throughput around 10 Mbits/s.
- **Fast Hash:** $r = 480$, $w = 170$ and $\log_2 \frac{n}{w} = 8$. This solution will be very fast (around 90 Mbits/s) with still a reasonable matrix size (20 Mbits).
- **Intermediate:** $r = 400$, $w = 85$ and $\log_2 \frac{n}{w} = 8$. This is in our opinion the best compromise, with reasonable hash length and matrix size and still a good efficiency (around 70 Mbits/s).

If looking only for a one-way function (no collision resistance) then we have the choice to either be faster, or have a smaller output.

- **Short One-Way:** $r = 240$, $w = 40$ and $\log_2 \frac{n}{w} = 8$. This solution has an output of only 240 bits and should work at around 70 Mbits/s.
- **Fast One-Way:** $r = 480$, $w = 160$ and $\log_2 \frac{n}{w} = 16$. This solution uses a very large matrix (4 Gbits) but should have a throughput above 200 Mbits/s.

6 Comparison to Existing Hash Functions

As stated at the beginning of Section 5, from a practical security point of view, our functions are somehow weaker than other existing functions: we will never be able to reach a security of $\mathcal{O}\left(2^{\frac{r}{2}}\right)$ against collision search. Accordingly, the output size of our functions will always have to be above 320 bits.

The description of one of our function will also always be much larger than that of other functions: the matrix should be included in the description and is always quite large when looking for fast hashing. However, as long as one uses

Table 2. Throughputs of some other hash functions, using the crypto++ library [5]

Algorithm	Mbits/s
MD5	1730
RIPEMD-160	420
SHA-1	544
SHA-512	90

parameters for which the matrix isn't of many Gigabits this shouldn't cause any problem.

From a speed point of view, our functions also seem much slower than existing functions. For instance, as seen in Table 2, using Wei Dai's crypto++ library, other hash functions are much faster than the 90 Mbits/s of **Fast Hash**. One should however take into account the fact that these 90 Mbits/s are obtained using a very basic C implementation, taking no advantage of the extended Pentium operations (MMX, SSE...).

The operations in FSB are however very simple: the only costly operations are binary XORs. Hence what will slow the process will mainly be memory access problems as the matrix $\mathcal{H}$ has no chance to fit in the machine's CPU cache. Without any fully optimized implementation of the algorithm it seems hard to estimate the place left for improvement.

However, depending of the use made of the hash function, the flexibility in the parameter choice of FSB can compensate this lack of speed. Imagine hashing for a 1024 bits RSA signature: you need to output a 1024 bits hash and at the same time do not require a security higher than 2^{80} as it would be higher than that of the RSA part of the signature. For such application, with $r = 1024$, one could use one of the following parameter sets:

a	security	$\log_2 \frac{n}{w}$	w	n	$\mathcal{N}_{XOR}$	matrix size
11	2^{85}	8	11655	2 983 680	129	3 Gbits
8	2^{113}	8	1942	497 152	137	485 Mbits

Still using our basic implementation this would yield throughputs around 70Mbits/s, which is not bad for a 1024 bits hash function. It seems that, for FSB, the throughput will depend more on the required security level than on the output hash size.

7 Conclusion

We have proposed a family of fast and provably secure hash functions. This construction enjoys some interesting features: both the block size of the hash function and the output size are completely scalable; the security depends directly of the output size and is truly exponential, it can hence be set to any desired level; the number of XORs used by FSB per input bit can be decreased to improve speed.

However, reaching very high output rates requires the use of a large matrix. This can be a limitation when trying to use FSB on memory constrained devices. On classical architectures this will only fix a maximum speed.

Another important point is the presence of weak instances of this hash function: it is clear that the matrix $\mathcal{H}$ can be chosen with bad properties. For instance, the all zero matrix will define a hash function with constant zero output. However, these bad instances only represent a completely negligible proportion of all the matrices and when choosing a matrix at random there is no real risk of choosing such a weak instance.

Also note that it is easy to introduce a trapdoor in a matrix by simply choosing one column to be the sum of some other columns of the matrix. This will then allow the person who generated the matrix to easily generate collisions. As stated by Preneel in [15], it is possible to avoid this problem if the matrix generation is reproducible by a user. The matrix could then simply be the output of a pseudo-random generator and this would solve both the problems of trapdoors and that of the huge matrix size. However, the security proof would no longer apply.

Finally, concerning the hash size/collision security ratio, this construction does not allow to have the usual ratio of 2, obtained when using a classical birthday paradox to find collisions. This can be changed by simply applying a final output transformation to the last hash: this transformation can further compress it to a size of twice the expected security against collision search. Further work has then to be done to study the required properties of this final transformation, both from a theoretical point of view, in order to keep the well founded security of these scheme, and from the practical point of view, in order to propose sound parameters.

References

1. D. Augot, M. Finiasz, and N. Sendrier. A fast provably secure cryptographic hash function. Cryptology ePrint Archive, 2003. `http://eprint.iacr.org/2003/230/`.
2. A. Barg. Complexity issues in coding theory. In V. S. Pless and W. C. Huffman, editors, *Handbook of Coding theory*, volume I, chapter 7, pages 649–754. North-Holland, 1998.
3. E. R. Berlekamp, R. J. McEliece, and H. C. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3), May 1978.
4. J.-S. Coron and A. Joux. Cryptanalysis of a provably secure cryptographic hash function. Cryptology ePrint Archive, 2004. http://eprint.iacr.org/2004/013/.
5. Wei Dai. Crypto++ library. http://www.eskimo.com/~weidai/.
6. I.B. Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO 89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–426. Springer-Verlag, 1989.
7. Y. Gurevich. Average case completeness. *Journal of Computer and System Sciences*, 42(3):346–398, 1991.
8. A. Joux and L. Granboulan. A practical attack against knapsack based hash functions. In Alfredo De Santis, editor, *Advances in Cryptology - Eurocrypt '94*, volume 950 of *Lecture Notes in Computer Science*, pages 58–66. Springer-Verlag, 1994.

9. L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986.
10. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Prog. Rep.,* Jet Prop. Lab., California Inst. Technol., Pasadena, CA, pages 114–116, January 1978.
11. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.
12. R. C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1990.
13. National Insitute of Standards and Technology. *FIPS Publication 180: Secure Hash Standard*, 1993.
14. H. Niederreiter. Knapsack-type crytosystems and algebraic coding theory. *Prob. Contr. Inform. Theory*, 15(2):157–166, 1986.
15. Bart Preneel. The state of cryptographic hash functions. In Ivan Damgård, editor, *Lectures on Data Security: Modern Cryptology in Theory and Practice*, volume 1561 of *Lecture Notes in Computer Science*, pages 158–182. Springer-Verlag, 1999.
16. R. L. Rivest. The MD4 message digest algorithm. In A.J. Menezes and S.A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311. Springer-Verlag, 1991.
17. P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388, 2004.
18. N. Sendrier. On the security of the McEliece public-key cryptosystem. In M. Blaum, P.G. Farrell, and H. van Tilborg, editors, *Information, Coding and Mathematics*, pages 141–163. Kluwer, 2002. Proceedings of Workshop honoring Prof. Bob McEliece on his 60th birthday.
19. D. Wagner. A generalized birthday problem. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–304. Springer-Verlag, 2002.
20. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions md4 and ripemd. In Ronald Cramer, editor, *Advances in Cryptology – Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18, Aarhus, Denmark, May 2005. Springer-Verlag.
21. Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology – Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35, Aarhus, Denmark, May 2005. Springer-Verlag.

A NP-Completeness Proofs

The most general problem we want to study concerning syndrome decoding with regular words is:

b-regular Syndrome Decoding (b-RSD)
Input: w binary matrices $\mathcal{H}_i$ of dimension $r \times n$ and a bit string $\mathcal{S}$ of length r.
Property: there exists a set of $b \times w'$ columns (with $0 < w' \leq w$), 0 or b columns in each $\mathcal{H}_i$, summing to $\mathcal{S}$.

Note that, in this problem, b is not an input parameter. The fact that for any value of b this problem is NP-complete is much stronger than simply saying that the problem where b is an instance parameter is NP-complete. This also means that there is not one, but an infinity of such problems (one for each value of b). However we consider them as a single problem as the proof is the same for all values of b.

The two following sub-problems are derived from the previous one. They correspond more precisely to the kind of instances that an attacker on the FSB hash function would need to solve.

Regular Syndrome Decoding (RSD)
Input: w matrices $\mathcal{H}_i$ of dimension $r \times n$ and a bit string $\mathcal{S}$ of length r.
Property: there exists a set of w columns, 1 per $\mathcal{H}_i$, summing to $\mathcal{S}$.

2-regular Null Syndrome Decoding (2-RNSD)
Input: w matrices $\mathcal{H}_i$ of dimension $r \times n$.
Property: there exists a set of $2 \times w'$ columns (with $0 < w' \leq w$), taking 0 or 2 columns in each $\mathcal{H}_i$ summing to 0.

It is easy to see that all of these problems are in NP. To prove that they are NP-complete we will use a reduction similar to the one given by Berlekamp, McEliece and van Tilborg for syndrome decoding [3]. We will use the following known NP-complete problem.

Three-Dimensional Matching (3DM)
Input: a subset $U \subseteq T \times T \times T$ where T is a finite set.
Property: there is a set $V \subseteq U$ such that $|V| = |T|$ and no two elements of V agree on any coordinate.

Let's study the following example: let $T = \{1, 2, 3\}$ and $|U| = 5$

$$\begin{aligned} U_1 &= (1, 2, 2) \\ U_2 &= (2, 2, 3) \\ U_3 &= (1, 3, 2) \\ U_4 &= (2, 1, 3) \\ U_5 &= (3, 3, 1) \end{aligned}$$

One can see that the set consisting of U_1,U_4 and U_5 verifies the property. However if you remove U_1 from U then no solution exist. In our case it is more convenient to represent an instance of this problem in another way: we associate a $3|T| \times |U|$ binary incidence matrix A to the instance. For the previous example it would give the matrix shown in Table 3.

A solution to the problem will then be a subset of $|T|$ columns summing to the all-1 column. Using this representation, we will now show that any instance of this problem can be reduced to solving an instance of RSD, hence proving that RSD is NP-complete.

Reductions of 3DM to RSD. Given an input $U \subseteq T \times T \times T$ of the 3DM problem, let A be the $3|T| \times |U|$ incidence matrix described above. For i from 1 to $|T|$ we take $\mathcal{H}_i = A$.

If we try to solve the RSD problem on these matrices with $w = |T|$ and $\mathcal{S} = (1, \ldots, 1)$ a solution will exist if and only if we are able to add $w \leq |T|$

Table 3. Incidence matrix corresponding to an instance of 3DM

	U_1	U_2	U_3	U_4	U_5
	122	223	132	213	331
1	1	0	1	0	0
2	0	1	0	1	0
3	0	0	0	0	1
1	0	0	0	1	0
2	1	1	0	0	0
3	0	0	1	0	1
1	0	0	0	0	1
2	1	0	1	0	0
3	0	1	0	1	0

columns of A (possibly many times the same one) and obtain a column of 1s. As all the columns of A contain only three 1s, the only way to have $3 \times |T|$ 1s at the end is that during the adding no two columns have a 1 on the same line (each time two columns have a 1 on the same line the final weight decreases by 2). Hence the $|T|$ chosen columns will form a suitable subset V for the 3DM problem.

This means that if we are able to give an answer to this RSD instance, we will be able to answer the 3DM instance we wanted to solve. Thus RSD is NP-complete.

Reduction of 3DM to *b*-RSD. This proof will be exactly the same as the one above. The input is the same, but this time we build the following matrix:

$$B = \begin{array}{|c|c|c|} \hline A & \multicolumn{2}{c|}{0} \\ \hline & A & \\ \hline \multicolumn{2}{|c|}{0} & A \\ \hline \end{array}$$

the block matrix with b times A on the diagonal

Now we take $\mathcal{H}_i = B$ and use $\mathcal{S} = (1, \ldots, 1)$. The same arguments as above apply here and prove that for any given value of b, if we are able to give an answer to this b-RSD instance, we will be able to answer the 3DM instance we wanted to solve. Hence, for any b, b-RSD is NP-complete.

Reduction of 3DM to 2-RNSD. We need to construct a matrix for which solving a 2-RNSD instance is equivalent to solving a given 3DM instance. A difficulty is that, this time, we can't choose $\mathcal{S} = (1, \ldots, 1)$ as this problem is restricted to the case $\mathcal{S} = 0$. For this reason we need to construct a somehow complicated matrix $\mathcal{H}$ which is the concatenation of the matrices $\mathcal{H}_i$ we will use. It is constructed as shown in Fig. 5.

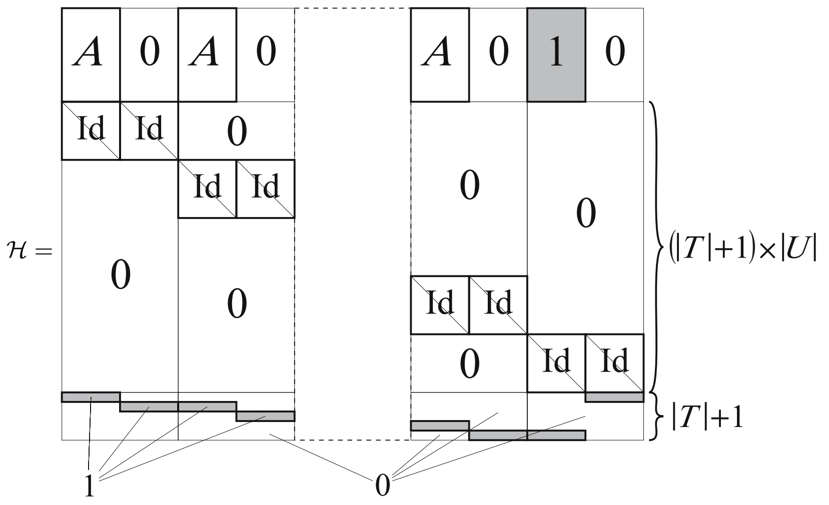


Fig. 5. The matrix used to reduce 3DM to 2-RNSD

This matrix is composed of three parts: the top part with the A matrices, the middle part with pairs of identity $|U| \times |U|$ matrices, and the bottom part with small lines of 1s.

The aim of this construction is to ensure that a solution to 2-RNSD on this matrix (with $w = |T| + 1$) exists if and only if one can add $|T|$ columns of A and a column of 1s to obtain 0. This is then equivalent to having a solution to the 3DM problem.

The top part of the matrix will be the part where the link to 3DM is placed: in the 2-RNSD problem you take 2 columns in some of the block, our aim is to take two columns in *each* block, and each time, one in the A sub-block and one in the 0 sub-block. The middle part ensures that when a solution chooses a column in $\mathcal{H}$ it has to choose the only other column having a 1 on the same line so that the final sum on this line is 0. This means that any time a column is chosen in one of the A sub-blocks, the "same" column is chosen in the 0 sub-block. Hence in the final $2w'$ columns, w' will be taken in the A sub-blocks (or the 1 sub-block) and w' in the 0 sub-blocks. You will then have a set of w' columns of A or 1 (not necessarily distinct) summing to 0. Finally, the bottom part of the matrix is there to ensure that if $w' > 0$ (as requested in the formulation of the problem) then $w' = w$. Indeed, each time you pick a column in the block number i, the middle part makes you have to pick one in the other half of the block, creating two ones in the final sum. To eliminate these ones the only way is to pick some columns in the blocks $i - 1$ and $i + 1$ and so on, until you pick some columns in all of the w blocks.

As a result, we see that solving an instance of 2-RNSD on $\mathcal{H}$ is equivalent to choosing $|T|$ columns in A (not necessarily different) all summing to 1. As in the previous proof, this concludes the reduction and 2-RNSD is now proven NP-complete.

It is interesting to note that instead of using 3DM we could directly have used RSD for this reduction. You simply replace the A matrices with the w blocks of the RSD instance you need to solve and instead of a matrix of 1s you put a matrix containing columns equal to $\mathcal{S}$. Then the reduction is also possible.

Optimization of Electronic First-Bid Sealed-Bid Auction Based on Homomorphic Secret Sharing

Kun Peng, Colin Boyd, and Ed Dawson

Information Security Institute,
Queensland University of Technology
{k.peng, c.boyd, e.dawson}@qut.edu.au
http://www.isrc.qut.edu.au

Abstract. Although secret sharing techniques have been applied to implement secure electronic sealed-bid auction for a long time, problems and attacks still exist in secret-sharing-based electronic sealed-bid auction schemes. In this paper, a new secret-sharing-based first-bid e-auction scheme is designed to achieve satisfactory properties and efficiency. Correctness and fairness of the new auction are based on hard computation problems and do not depend on any trust. Complete bid privacy based on a threshold trust is achieved in the new scheme. Attacks to existing secret-sharing-based sealed-bid e-auction schemes are prevented.

1 Introduction

The first secure electronic sealed-bid auction scheme [3] is based on threshold secret sharing. Since then, more secret-sharing-based sealed-bid e-auction schemes [4,6,5,10] have been proposed. Most of them [4,6,10] are supposed to support first-bid sealed-bid e-auction. However as will be shown many security problems exist in these auction schemes and they are vulnerable to various attacks. The newest and most advanced of them, [10], pointed out lack of secret sharing verification and vulnerability to three attacks in the previous secret-sharing-based sealed-bid e-auctions. However, the countermeasures in [10] cannot completely prevent these three attacks. In this paper, drawbacks of the previous secret-sharing-based sealed-bid e-auction schemes are listed and analysed. Then a new secret-sharing-based sealed-bid auction scheme is proposed, which can implement secure and efficient first-bid sealed-bid e-auction. Several attacks in the existing secret-sharing-based sealed-bid e-auction schemes are prevented in the new scheme.

2 Requirements and Related Work

Auction is a useful tool to distribute resources. The principle of auction is to sell goods at the highest possible price. Sealed-bid auction usually contains four phases: preparation phase, bidding phase, bid opening phase and winner determination phase.

1. In the preparation phase, the auction system is set up and the auction rule is published.
2. In the bidding phase, every bidder submits a sealed bid through a communication network.
3. In the bid opening phase, the bids are opened to determine the winning price.
4. In the winner determination phase, the winner is identified.

The following properties are often desired in sealed-bid auction.

1. **Correctness**: The auction result is determined strictly according to the auction rule. For example, if first bid auction is run, the bidder with the highest bid wins and pays the highest bid.
2. **Bid confidentiality**: Each bid remains confidential to anyone other than the bidder himself before the bid opening phase starts.
3. **Fairness**: No bidder can take advantage of other bidders (e.g. recover other bids and choose or change his own bids according to other bids).
4. **Unchangeability**: Any bidder, especially the winner, cannot change or deny his bid after it is submitted.
5. **Public verifiability**: Correctness of the auction (including validity of the bids, correctness of bid opening and correctness of winner identification) must be publicly verifiable.
6. **Bid Privacy**: Confidentiality of the losing bids must be still retained after the auction finishes. Strictly speaking, no information about any losing bid is revealed except what can be deduced from the auction result.
7. **Robustness**: The auction can still run properly in abnormal situations like existence of invalid bid.

The commonly used auction rules in sealed-bid auctions include first bid auction and Vickrey auction. In a first bid auction, the bidder with the highest bid wins and pays the highest bid. In a Vickrey auction, the bidder with the highest bid wins and pays the second highest bid. Another popular rule, the i^{th} bid auction [5] is a multiple-item version of first bid auction or Vickrey auction.

In a secure auction scheme, secrecy of the bid is very important. Usually, bid confidentiality must be achieved without any trust on the auctioneers, as loss of confidentiality is fatal to fairness of the auction. If a bidder can collude with some auctioneers to know other bids before submitting his own bid, he can win at a price as low as possible in a first bid auction, which violates the principle and fairness of auction. On the other hand, bid privacy can be based on some trust, like a threshold trust on the auctioneers as breach of a bidder's personal privacy is not so serious and is tolerable in some cases. Implementation of bid privacy is rule-dependent. Although Vickery auction is preferred in many applications, it is difficult to achieve bid privacy in Vickrey auction. As the winner's bid and the identity of the bidder submitting the winning bid must be kept secret as required in bid privacy, there is no practical method to achieve bid privacy in Vickrey auction. As bid privacy is required in this paper, we focus on first-bid auction.

Except [3], all the secret-sharing-based sealed-bid e-auction schemes employ one-choice-per-price strategy. Under this strategy, the price space (containing all the biddable prices) is much smaller than the input domain of the sealing function and each bidder must make a choice (indicating willingness or unwillingness to pay) at every biddable price to form his bidding vector. If a bidder is willing to pay a price, he chooses a non-zero integer standing for "YES" as his choice at that price. If a bidder is unwilling to pay a price, he chooses zero standing for "NO" as his choice at that price. The bidders seal their bidding vectors (including choices at all the biddable prices) and submit the sealed bidding vectors in the bidding phase. These sealed-bid e-auction schemes also employ additive homomorphic secret sharing and binary search. The bidders use additive homomorphic secret sharing (e.g. Shamir's secret sharing [12] and its variants) to seal their bidding choices. Then a binary search for the winning price is performed along a binary route among the biddable prices. In the search, the auctioneers exploit additive homomorphism of the sharing function (the summed shares of all the choices at a price can reconstruct the sum of the choices at that price) to implement bid opening at every price on the binary searching route until finally the winning price is met.

Among the existing secret-sharing-based first-bid e-auction schemes [3,4,6,10], the most recent and advanced one is [10]. The auction scheme in [3] is simple, but does not support bid privacy. Secret bid sharing in [3,4,6] is not verifiable, so the bids are changeable and a bidder can collude with an auctioneer to compromise correctness and fairness. Besides lack of verifiability [10] points out three attacks to the previous schemes [4,6], ABC (auctioneer-bidder collusion) attack, BBC (bidder-bidder collusion) attack and dispute attack. In an ABC attack, some auctioneers collude with a bidder to compromise correctness or fairness. In a BBC attack, some bidders collude to compromise correctness or fairness. In a dispute attack, an auctioneer accuses a bidder of submitting an invalid (encrypted) choice share and the bidder cannot prove his innocence without revealing the share. However, the auction scheme in [10] cannot completely prevent these three attacks.

The auction scheme in [10] prevented an ABC attack in [4,6]: an auctioneer helps a bidder to change his bid after submitting it. However, [10] is vulnerable to another ABC attack in first-bid auction. As bid sealing depends on threshold secret sharing, any submitted sealed bid can be opened before the bid opening phase if the number of malicious auctioneers is over the sharing threshold. These malicious auctioneers then can reveal the opened bids to a waiting colluding bidder, who can bid just higher than the submitted bids and win at a price as low as possible. This attack is an ABC attack and definitely compromises fairness of the auction. Although a threshold trust on the auctioneers is assumed in [10] and this ABC attack does not exist under the threshold trust, this threshold trust assumption is too strong for correctness and fairness of the auction. It is appropriate to base less important properties like bid privacy on the threshold trust assumption. However, as stated before, bid confidentiality must be achieved without any trust on the auctioneers as it affects correctness and fairness of the auction. So this ABC attack against correctness and fairness must be prevented without any assumption on the auctioneers.

The existing secret-sharing-based sealed-bid e-auction schemes are vulnerable to BBC attack as well. For example, three colluding bidders B_1, B_2 and B_3 may perform the following attack against first bid auction where in a bidding choice no-zero integer Y and 0 stand for "YES" and "NO" respectively.

– B_1, B_2 and B_3 estimate that the other bidders' bids are lower than p_μ while their own evaluation is p_ν, which is higher than p_μ. They try to win the auction and pay as low as possible.
– B_1 bids Y at prices no higher than p_μ and zero at other prices; B_2 bids Y at prices no higher than p_ν and zero at other prices; B_3 bids $-Y$ at prices higher than p_μ but no higher than p_ν and zero at other prices.
– If all other bidder submits a bid lower than p_μ as expected, the sum of choices at p_μ is non-zero and the sum of choices at prices higher than p_μ is 0. So p_μ is the winning price and there is a tie between B_1 and B_2. One of them gives up and the other wins at p_μ.
– If other bidders' highest bid, p_H is no lower than p_μ but lower than p_ν, the sum of choices at p_H is larger than zero and the sum of choices at prices higher than p_H is 0. So some other bidder wins the auction at p_H together with B_2. B_2 disputes the tie and publishes his bid to win the auction at p_ν.
– If other bidders' highest bid is p_ν, the sum of choices at p_ν is larger than zero and the sum of choices at prices higher than p_ν is 0. So some other bidder draws with B_2 at p_ν. B_2 still has a chance to win the auction in the following tie-breaking operation.

With this attack, either B_1 or B_2 win unless another bidder submits a bid higher than the attackers' evaluation. The attackers can pay a price lower than their evaluation if the other bids are as low as the attackers expect.

It is pointed out in [10] that the previous secret-sharing-based e-auction schemes [3,4,6,5] are not publicly verifiable when bid privacy must be retained. As a result of lack of pubic verifiability, these schemes cannot deal with dispute between bidders and auctioneers, so are vulnerable to the dispute attack. To prevent the dispute attack, [10] suggests to use publicly verifiable secret sharing (PVSS) to distribute the bids. A PVSS protocol based on Bao's proof of equality of logarithms in different cyclic groups with different orders [1] is proposed in [10]. However, Bao's proof is neither specially sound nor zero knowledge. Bao only used it in a special verifiable encryption scheme, where he believes soundness and ZK property of the proof is not necessary. Application of Bao's proof in [10] is not appropriate. The PVSS in [10] cannot prevent the dispute attack as invalid bid can pass its verification. Moreover, the PVSS reveals some information about the bids.

To protect bid confidentiality and privacy when the bidding choices are in a small set, information-theoretically hiding secret sharing scheme proposed by Pedersen [9] is employed in [10] to share the bidding choices, whose computational and communication cost is twice as high as a computationally hiding verifiable secret sharing scheme like [8]. However, as will be shown later in the new auction scheme in this paper a computationally hiding verifiable secret sharing is enough to protect bid confidentiality and privacy if the auction protocol is well designed. Although information-theoretically hiding property is achieved in the bid sharing in [10] at a high cost, bids in that scheme are not information-theoretically confidential and not even semantically confidential as Paillier encryption is simplified in [10] to lose semantic security. In addition, bid privacy is not complete in [10]. At every price on the binary searching route, the number of "YES" choices is revealed.

In this paper, a new secret-sharing-based first-bid e-auction is designed, which can prevent the three attacks and achieve complete bid privacy more efficiently.

3 Pedersen's Verifiable Secret Sharing

Since Shamir proposed the first threshold secret sharing scheme [12], many threshold secret sharing techniques have appeared. Using these techniques, a secret holder can share a secret among multiple share holders. The secret can be recovered if the number of cooperating share holders is over a certain threshold, T. If the secret holder is not trusted, there must be a mechanism the share holders can use to verify that they get a set of valid shares of a unique secret. This requirement is very important for the robustness of applications like auctions. Secret sharing with this mechanism is called VSS (verifiable secret sharing). Shamir's secret sharing was extended by Pedersen to be verifiable as follows [8].

1. G is the subgroup of Z_p^* with order q where p and q are large primes such that q divides $p-1$. Integer g is a generator of G.
2. A builds a polynomial $f(x) = \sum_{j=0}^{T} a_j x^j$ where $a_0 = s$ and a_j for $j = 1, 2, \ldots, T$ are random integers.
3. A publishes $E_j = g^{a_j}$ for $j = 0, 1, \ldots, T$.
4. A sends $s_i = f(i)$ as a share to share holder P_i.
5. P_i verifies $g^{s_i} = \prod_{j=0}^{T} E_j^{i^j}$. If the verification is passed, P_i can be sure that s_i is the i^{th} share of $\log_g E_0$.
6. If at least $T+1$ share holders get correct shares, $\log_g E_0$ can be recovered by them corporately.

In this paper, Pedersen's verifiable secret sharing will be employed, which has the following three properties.

– Correctness: if the secret holder follows the VSS protocol, he can share his secret such that each share can pass the verification.
– Soundness: if the verification is passed, any share set containing more than T shares can be used to recover secret $\log_g E_0$.
– Homomorphism: if multiple secrets are shared among the same sets of share holders, they can sum up the shares to recover the sum of the secrets.

4 The New Auction Scheme

The basic structure of a secret-sharing-based sealed-bid e-auction is inherited in this new scheme. Like in other secret-sharing-based sealed-bid e-auction schemes, the bidders share their bids among the auctioneers, who employ homomorphic bid opening and binary search to determine the winning price. However, in the new auction schemes, certain measures are taken to prevent the attacks and overcome the drawbacks in the existing secret-sharing-based sealed-bid e-auction schemes. Two rounds of communication are employed between the bidders and the auctioneers, while only one round of communication is employed in the existing schemes. In the first round the bidders commit to their bids and publish the commitments. The committing function is information-theoretically hiding, such that it is impossible for anyone to recover any bid from the commitments. The committing function is computationally binding, such that to find two different ways to open the commitments is as hard as the discrete logarithm problem. In the second round the bidders share the bid opening information among the auctioneers through an additive homomorphic VSS mechanism, so that the auctioneers can cooperate to recover sum of the bidding choices. Hiding property of the committing function prevents ABC attack, while binding property of the committing function guarantees unchangeability. The auctioneers randomize the bidding choices before they are summed up, so that BBC attack is prevented. The verifiable secret sharing in [8] is employed in the new scheme, which is additive homomorphic and efficient. A dispute-settling function based on that VSS technique and verifiable encryption is designed to settle dispute on validity of encrypted shares. As the bidding choices are randomized before they are summed up, no information about the losing bids is revealed although the sum of the bidding choices is published at the prices on the binary searching route.

Suppose there are w biddable prices $p_1, p_2, \ldots, p_w$ in decreasing order, n bidders $B_1, B_2, \ldots, B_n$ and m auctioneers $A_1, A_2, \ldots, A_m$. The auction protocol is as follows.

1. **Preparation phase**
   A bulletin board is set up as a broadcast communication channel. Each A_j establishes his Paillier encryption [7]) algorithm with public key N_j (product of two secret large primes) and g_j (whose order is a multiple of N_j), message space Z_{N_j}, multiplicative modulus N_j^2, encryption function $E_j(x) = g_j^x r^{N_j} \bmod N_j^2$ and a corresponding decryption function $D_j()$. A_j publishes on the bulletin board his encryption function and public key for $j = 1, 2, \ldots, m$. Large primes p and q are chosen such that q is a factor of $p - 1$ and $nq^2 < N_j$ for $j = 1, 2, \ldots, m$. Cyclic group G contains all the quadratic-residues in Z_p^* and has an order q. Random primes f, g and h are chosen such that $\log_g f$ and $\log_h g$ are unknown. The bid committing function is $Com(x) = f^x g^r \bmod p$ where x is a bidding choice in Z_q and r is a random integer in Z_q. A sharing threshold parameter T smaller than m is chosen. System parameters p, q, f, g, h, T and N_j for $j = 1, 2, \ldots, m$ are published on the bulletin board.
2. **Bidding phase**
   Each bidder B_i selects his bidding vector $(b_{i,1}, b_{i,2}, \ldots, b_{i,w})$ as his choices at $p_1, p_2, \ldots, p_w$ where $b_{i,l} \in Z_q$ for $l = 1, 2, \ldots, w$. If he is willing to pay p_l, $b_{i,l}$ is a random non-zero integer modulo q; if he is unwilling to pay p_l,

$b_{i,l} = 0$. Then he signs and publishes $c_{i,l} = Com(b_{i,l}) = f^{b_{i,l}} g^{r_{i,l}} \bmod p$ for $l = 1, 2, \ldots, w$ on the bulletin board where $r_{i,l}$ is randomly chosen from Z_q.

3. **Bid opening phase**
   (a) Bid randomization
   Each auctioneer A_j publishes a commitment (e.g. one-way hash function) of random integer $R_{j,i,l}$ from Z_q for $i = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, w$. After all the commitments have been published, the auctioneers publish $R_{j,i,l}$ for $j = 1, 2, \ldots, m$ as randomizing factors of $b_{i,l}$ on the bulletin board.
   (b) Secret sharing
   Each B_i calculates $R_{i,l} = \sum_{j=1}^{m} R_{j,i,l} \bmod q$. Then he calculates $s_{i,l} = r_{i,l} R_{i,l} \bmod q$ as his secret at p_l for $l = 1, 2, \ldots, w$. B_i chooses polynomials $F_{i,l}(x) = \sum_{k=0}^{T} a_{i,l,k} x^k \bmod q$ for $l = 1, 2, \ldots, w$ where $a_{i,l,0} = s_{i,l}$ and $a_{i,l,k}$ for $k = 1, 2, \ldots, T$ are randomly chosen. B_i publishes encrypted shares $S_{i,l,j} = E_j(F_{i,l}(j)) = g_j^{F_{i,l}(j)} t_{i,l,j}^{N_j} \bmod N_j^2$ for $l = 1, 2, \ldots, w$ and $j = 1, 2, \ldots, m$ on the bulletin board where $t_{i,l,j}$ is randomly chosen from $Z_{N_j}^*$. B_i publishes sharing commitments $C_{i,l,k} = h^{a_{i,l,k}} \bmod p$ for $l = 1, 2, \ldots, w$ and $k = 0, 1, \ldots, T$ on the bulletin board.
   (c) Binary search
   The auctioneers cooperate to perform a binary search. At a price p_l on the searching route, the following operations are performed.
      i. Share verification
      Each A_j calculates his summed shares $v_{j,l} = D_j(\prod_{i=1}^{n} S_{i,l,j} \bmod N_j^2)$ and the corresponding commitments $u_{l,k} = \prod_{i=1}^{n} C_{i,l,k} \bmod p$ for $k = 0, 1, \ldots, T$. He then verifies $h^{v_{j,l}} = \prod_{k=0}^{T} u_{l,k}^{j^k} \bmod p$. If the verification is passed, he goes on to next step. Otherwise, he verifies $h^{D_j(S_{i,l,j})} = \prod_{k=0}^{T} C_{i,l,k}^{j^k} \bmod p$ for $i = 1, 2, \ldots, n$ and will meet at least one failed verification. If the verification fails when $i = I$, A_j accuses bidder B_I of submitting an invalid encrypted share $S_{I,l,j}$. If B_I disputes on the accusation, the following dispute settling procedure is used. A_j publishes $z_{I,l,j} = D(S_{I,l,j})$ such that anyone can verify $h^{z_{I,l,j}} \neq \prod_{k=0}^{T} C_{I,l,k}^{j^k} \bmod p$. If $h^{z_{I,l,j}} \neq \prod_{k=0}^{T} C_{I,l,k}^{j^k} \bmod p$, B_I has to publish $t_{I,l,j}$ and proves his knowledge of $\log_{g_j}(S_{I,l,j}/t_{I,l,j}^{N_j})$ using the zero knowledge proof of knowledge of logarithm in [11][1]. B_I asks the auctioneers to verify his proof and $S_{I,l,j} \neq g_j^{z_{I,l,j}} t_{I,l,j}^{N_j} \bmod N_j^2$. If

$$h^{z_{I,l,j}} = \prod_{k=0}^{T} C_{I,l,k}^{j^k} \bmod p \vee$$
$$(S_{I,l,j} \neq g_j^{z_{I,l,j}} t_{I,l,j}^{N_j} \bmod N_j^2 \ \wedge \ B_I\text{'s proof is correct})$$

[1] Although the parameter setting in [11] is a little different from the parameter setting in Paillier encryption (When [11] was proposed, Paillier encryption had not appeared), the proof protocol in [11] can be applied here without compromising its correctness, soundness or zero knowledge property.

the accusation against B_I is wrong and A_j is removed. Otherwise, B_I is removed from the auction and may be punished; share verification is run again.

ii. Homomorphic secret recovery
Each A_j publishes $v_{j,l}$, whose validity can be verified by anyone against $C_{i,l,k}$ for $i = 1, 2, \dots, n$ and $k = 0, 1, \dots, T$. If at least $T+1$ summed shares are correct, the summed secret can be recovered. For simplicity, suppose the first $T+1$ summed shares are correct, then the summed secret is recovered: $d_l = \sum_{i=1}^{n} s_{i,l} = \sum_{j=1}^{T+1} v_{j,l}^{x_j} \bmod q$ where $x_j = \prod_{1 \le k \le t+1, k \ne j} \frac{k}{k-j} \bmod q$.

iii. Homomorphic bid opening
Equation $\prod_{i=1}^{n} c_{i,l}^{R_{i,l}} = g^{d_l} \bmod p$ is tested. If this equation is correct, the sum of the randomized bidding choices at p_l is zero, the binary search at p_j ends negatively and the search goes down. If this equation is incorrect, the sum of randomized bidding choices at p_l is not zero, the binary search at p_j ends positively and the search goes up.

In the end of the binary search, the winning price is found.

4. **Winner identification phase**
Suppose the winning price is p_L. Decrypted shares $d_{i,L,j} = D_j(S_{i,L,j})$ for $i = 1, 2, \dots, n$ are published. $h^{d_{i,L,j}} = \prod_{k=0}^{T} C_{i,L,k}^{j^k} \bmod p$ is verified for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$. If any bidder's secret is found to be incorrectly shared, he is removed from the auction and may be punished. If he disputes, the dispute can be solved like in Step 3(c)i. If at least $T+1$ correct shares can be found for B_i, his secret $d_{i,L}$ can be recovered: $d_{i,L} = \sum_{j=1}^{T+1} d_{i,L,j}^{x_j} \bmod p$ (For simplicity, assume the first $T+1$ shares are correct). Then equation $c_{i,L}^{R_{i,L}} = g^{d_{i,L}} \bmod p$ is tested for $i = 1, 2, \dots, n$. Only when $c_{i,L}^{R_{i,L}} \ne g^{d_{i,L}} \bmod p$, is B_i a winner. Suppose $c_{I,L}^{R_{I,L}} \ne g^{d_{I,L}} \bmod p$. Then B_I must prove that he is really a winner by proving knowledge of $\log_f(c_{I,L}^{R_{I,L}}/g^{d_{I,L}})$ using the zero knowledge proof of knowledge of logarithm in [11]. Any B_I failing to give this proof is a cheater and punished. The winner's signature is verified and his identity is published. If there is more than one winner, a new auction is run among the winners.

Table 1. Comparison of homomorphic secret-sharing-based first-bid auction schemes

Operation	[4,6,10]	New auction
the first round of communication	share bidding choice $b_{i,l}$	commit bidding choice $b_{i,l}$ in $c_{i,l} = f^{b_{i,l}} g^{r_{i,l}}$
the second round of communication	non-existent	randomize $r_{i,l}$ into $s_{i,l} = r_{i,l} R_{i,l}$ and share $s_{i,l}$
bid opening	recover $\sum_{i=1}^{n} b_{i,l}$ and test whether $\sum_{i=1}^{n} b_{i,l} > 0$	recover $d_l = \sum_{i=1}^{n} s_{i,l}$ and test whether $\prod_{i=1}^{n} c_{i,l}^{R_{i,l}} = g^{d_l}$
dispute settlement	non-existent or vulnerable	solved by VSS and verifiable encryption

In Table 1, the new secret-sharing-based first-bid auction scheme is compared against the existing homomorphic secret-sharing-based first-bid auction schemes [4,6,10]. It is clear that in the new scheme, two-round communication and commitment prevent the ABC attack; the randomization prevents the BBC attack and strenghens bid privacy; the dispute settling procedure prevents the dispute attack.

5 Analysis

Security and efficiency of the new auction scheme is analysed in this section. Readers can check that if the bidders and auctioneers follow the protocol, the auction outputs a correct result. Note that although two different kinds of additive modulus p and N_j are used in the protocol, no modulus conflict happens as $r_{i,l}$, $R_{i,l}$ are chosen from Z_q and $nq^2 < N_j$.

5.1 Security Analysis

In the following, it is demonstrated that the auction is correct as long as at least one auctioneer is honest.

Theorem 1. *The auction protocol is correct with an overwhelmingly large probability if at least one auctioneer is honest. More precisely, the bidder with the highest bid wins with an overwhelmingly large probability if at least one auctioneer is honest.*

To prove this theorem, the following three lemmas must be proved first.

Lemma 1. *If* $\sum_{i=1}^{n} y_i s_i = 0 \bmod q$ *with a probability larger than* $1/q$ *for random* $s_1, s_2, \ldots, s_n$ *from* Z_q*, then* $y_i = 0 \bmod q$ *for* $i = 1, 2, \ldots, n$.

Proof: Given any integer k in $\{1, 2, \ldots, n\}$, there must exist integers $s_1, s_2, \ldots,$ $s_{k-1}, s_{k+1}, \ldots, s_n$ in Z_q and two different integers s_k and $\hat{s}_k$ in Z_q such that the following two equations are correct.

$$\sum_{i=1}^{n} y_i s_i = 0 \bmod q \tag{1}$$

$$(\sum_{i=1}^{k-1} y_i s_i) + y_k \hat{s}_k + \sum_{i=k+1}^{n} y_i s_i = 0 \bmod q \tag{2}$$

Otherwise, for any $s_1, s_2, \ldots, s_{k-1}, s_{k+1}, \ldots, s_n$ there is at most one s_k to satisfy equation $\sum_{i=1}^{n} y_i s_i = 0 \bmod q$. This deduction implies among the q^n possible combinations of $s_1, s_2, \ldots, s_n$, equation $\sum_{i=1}^{n} y_i s_i = 0 \bmod q$ is correct for at most q^{n-1} combinations. This conclusion leads to a contradiction: given random integers s_i from Z_q for $i = 1, 2, \ldots, n$, equation $\sum_{i=1}^{n} y_i s_i = 0 \bmod q$ is correct with a probability no larger than $1/q$.

Subtracting (2) from (1) yields

$$y_k(s_k - \hat{s}_k) = 0 \bmod q$$

Note that $s_k - \hat{s}_k \neq 0 \bmod q$ as $s_k \neq \hat{s}_k \bmod q$. So, $y_k = 0 \bmod q$. Note that k can be any integer in $\{1, 2, \ldots, n\}$. Therefore $y_i = 0 \bmod q$ for $i = 1, 2, \ldots, n$. □

Lemma 2. *When the binary search at a price p_l ends negatively, $b_{i,l} = 0$ for $i = 1, 2, \ldots, n$ with an overwhelmingly large probability if at least one auctioneer is honest where $b_{i,l}$ is B_i's choice at p_l and committed in $c_{i,l}$.*

Proof: That the binary search at a price p_l ends negatively implies

$$\prod_{i=1}^{n} c_{i,l}^{R_{i,l}} = g^{d_l} \bmod p$$

where d_l is the summed secret recovered at p_l. So

$$\prod_{i=1}^{n} (f^{b_{i,l}} g^{r_{i,l}})^{R_{i,l}} = g^{d_l} \bmod p$$

Namely

$$f^{\sum_{i=1}^{n} R_{i,l} b_{i,l}} g^{\sum_{i=1}^{n} R_{i,l} r_{i,l}} = g^{d_l} \bmod p$$

Note that $b_{i,l}$ is committed in $c_{i,l} = f^{b_{i,l}} g^{r_{i,l}}$ by B_i and d_l is recovered from the shares from the bidders. So the bidders can cooperate to find $\sum_{i=1}^{n} R_{i,l} b_{i,l}$, $\sum_{i=1}^{n} R_{i,l} r_{i,l}$ and d_l in polynomial time.

So, if $\sum_{i=1}^{n} R_{i,l} b_{i,l} \neq 0$, the bidders can cooperate to find in polynomial time

$$\log_g f = (d_l - \sum_{i=1}^{n} R_{i,l} r_{i,l}) / \sum_{i=1}^{n} R_{i,l} b_{i,l},$$

which is contradictory to the assumption that $\log_g f$ is unknown and the discrete logarithm problem is hard to solve. So, $\sum_{i=1}^{n} R_{i,l} b_{i,l} = 0$. Note that $R_{i,l}$ for $i = 1, 2, \ldots, n$ are random integers in Z_q as they are corporately chosen by the auctioneers, at least one of which is honest. Therefore, according to Lemma 1. $b_{i,l} = 0$ for $i = 1, 2, \ldots, n$ with an overwhelmingly large probability. □

Lemma 3 guarantees that no bidder can change a “YES bid into a “NO bid.

Lemma 3. *If the binary search at a price p_l ends positively, then there exists I in $\{1, 2, \ldots, n\}$ such that one of the following two statements is true: 1) $b_{I,l} \neq 0$; 2) $b_{I,l} = 0$ but B_I cannot find $\log_f(c_{I,l}^{R_{I,l}} / g^{d_{I,l}})$ in polynomial time.*

Proof: That the binary search at a price p_l ends positively implies

$$\prod_{i=1}^{n} c_{i,l}^{R_{i,l}} \neq g^{d_l} \bmod p$$

where d_l is the summed secret recovered at p_l.

Soundness and homomorphism of the employed VSS [8] guarantees that

$$h^{d_l} = u_{l,0} = \prod_{i=1}^{n} C_{i,l,0} \bmod p$$

So,

$$\prod_{i=1}^{n} c_{i,l}^{R_{i,l}} \neq g^{\log_h \prod_{i=1}^{n} C_{i,l,0}} \bmod p$$

Namely,

$$\prod_{i=1}^{n} c_{i,l}^{R_{i,l}} \neq g^{\sum_{i=1}^{n} \log_h C_{i,l,0}} \bmod p,$$

which implies

$$\prod_{i=1}^{n} c_{i,l}^{R_{i,l}} \neq \prod_{i=1}^{n} g^{\log_h C_{i,l,0}} \bmod p$$

So there must exists integer I such that $1 \leq I \leq n$ and

$$c_{I,l}^{R_{I,l}} \neq g^{\log_h C_{I,l,0}} \bmod p$$

Suppose $g^{\log_h C_{I,l,0}}/(c_{I,l}^{R_{I,l}}) = f^{e_I}$, then $e_I \neq 0 \bmod q$. So

$$c_{I,l}^{R_{I,l}} = f^{e_I} g^{\log_h C_{I,l,0}} \bmod p$$

Namely

$$(f^{b_{I,l}} g^{r_{I,l}})^{R_{I,l}} = f^{e_I} g^{\log_h C_{I,l,0}} \bmod p$$

where $b_{i,l}$ is B_i's choice at p_l, which is committed in $c_{i,l}$.

Note that B_I knows $b_{I,l}$ and $r_{I,l}$ as he committed to $b_{I,l}$ as $c_{I,l} = f^{b_{I,l}} g^{r_{I,l}}$; B_I can find $\log_h C_{I,l,0}$ in polynomial time as his shares at p_j enable anyone to calculates $\log_h C_{I,l,0}$ in polynomial time. So, if B_I can find e_I in polynomial time, he can find $\log_g f = (\log_h C_{I,l,0} - r_{I,l} R_{I,l})/(b_{I,l} R_{I,l} - e_I)$ in polynomial time. So, when $b_{I,l} = 0$ either a contradiction to the assumption that $\log_g f$ is unknown and the discrete logarithm problem is hard to solve is found or B_I cannot find $e_I = \log_f(c_{I,l}^{R_{I,l}}/g^{d_{I,L}})$. Therefore, there exists I in $\{1, 2, \ldots, n\}$ such that one of the following two statements is true.

- $b_{I,l} \neq 0$;
- $b_{I,l} = 0$ but B_I cannot find $\log_f(c_{I,l}^{R_{I,l}}/g^{d_{I,l}})$ in polynomial time. □

Proof of Theorem 1:
Lemma 2 and Lemma 3 guarantee that if there is at least one honest auctioneer

- when a bidder submits a "YES" choice at a price, he can open it as a "NO" choice with an overwhelmingly small probability;
- when a bidder b_i submits a "NO" choice at a price p_l but opens it as a "YES" choice, he cannot find $\log_f(c_{i,l}^{R_{i,l}}/g^{d_{i,l}})$ in polynomial time.

So the binary search guarantees that if there is at least one honest auctioneer

- when a bidder submitted a “YES” choice at a price on the searching route, the search always go upwards at that price with an overwhelmingly large probability;
- when a bidder b_i submitted a “NO” choice at a price p_l on the searching route, either the search always go downwards at p_l or he cannot find $\log_f(c_{i,l}^{R_{i,l}}/g^{d_{i,l}})$ in polynomial time.

So when winning price p_L is determined in the bid opening phase,

- there is no “YES” choice at higher prices with an overwhelmingly large probability if there is at least one honest auctioneer;
- if there is at least one honest auctioneer, then at the winning price
  - either there is at least one “YES” choice,
  - or a bidder B_i submits “NO” choice at p_L, open it as “YES”, but cannot find $\log_f(c_{i,L}^{R_{i,L}}/g^{d_{i,L}})$ in polynomial time.

  Note that in the winner identification phase, any bidder B_I opening his choice as “Yes” at the winning price must prove that he is really a winner by proving knowledge of $\log_f(c_{I,L}^{R_{I,L}}/g^{d_{I,L}})$. So in the winner identification phase either some winner or some cheating bidder is identified with an overwhelmingly large probability if there is at least one honest auctioneer. If a winner is found, the auction ends correctly. If only cheating bidder(s) is found, the cheating bidder is removed and the auction runs again. Finally a correct winner can be definitely found when all the cheating bidders have been removed. □

If a penalty to identified cheating bidders is applied, the bidders will be deterred from cheating and re-running can be avoided. If no strong penalty is available and the re-running mechanism after finding a cheating bidder is not appropriate in some special applications, the auction protocol can be slightly modified so that the winning price found in the bid opening phase is always correct and a real winner can always be found at the winning price. Only a simple additional operation is needed in the modification: each bidder has to prove that his submitted bid hidden by $Com()$ is consistent with the secret shares provided by him in the bid opening phase. In the proof B_i shows that at each biddable price p_l he knows two secrets $b_{i,l}$ and $r_{i,l}$ such that $c_{i,l} = f^{b_{i,l}}g^{r_{i,l}}$ and $C_{i,l,0} = h^{r_{i,l}R_{i,l}}$ without revealing $b_{i,l}$ or $r_{i,l}$. This proof can be built on ZK proof of knowledge of logarithm [11] and ZK proof of equality of logarithms [2]. With this modification Lemma 3 can be modified to Lemma 4, which is simpler.

Lemma 4. *If the binary search at a price p_l ends positively, then there exists I in $\{1, 2, \ldots, n\}$ such that $b_{I,l} \neq 0$.*

The proof of Lemma 4 is simpler than that of Lemma 3, so is not provided here. With this modification and Lemma 4, Theorem 1 can be proved more easily, winner identification becomes simpler and rerunning can be avoided. However, bidding becomes less efficient with additional $O(nw)$ ZK proof and verification operations. In most cases, we believe that punishment can deter the bidders from cheating and rerunning can be avoided. So usually, this additional proof is not adopted for the sake of efficiency, which is assumed in efficiency analysis later.

Table 2. Property comparison

Auction schemes	Correct- -ness	Bid confi- -dentiality	Fairness	Unchan- -geability	Public verifiability	Bid privacy	Robust- -ness
[3]	Vulnerable to attacks	Trust -de -pendent	Trust -de -pendent	No	Yes	No	Yes
[4] [6]	Vulnerable to attacks	Trust de- -pendent	Trust de- -pendent	No	No	Trust de- -pendent	No
[10]	Vulnerable to attacks	Trust de- -pendent	Trust de- -pendent	Yes	Yes	Trust de- -pendent	No
New auction	Yes	Yes	Yes	Yes	Yes	Trust de- -pendent	Yes

As the commitment function $Com()$ is information-theoretically hiding, bid confidentiality is information-theoretically achieved and the ABC attack is information-theoretically prevented. Binding of $Com()$ and usage of digital signature guarantees unchangeability. As the bidding choices are randomized before they are summed up, the BBC attack can be prevented if at least one auctioneer is honest. The employed VSS [8] and the new dispute settling procedure[2] in Step 3(c)i can solve the dispute attack. With these three attacks prevented, the new auction protocol is fair and robust. Every operation in the auction protocol is publicly verifiable. If the number of malicious auctioneers is not over the sharing threshold, bid privacy can be achieved.

A property comparison of the secret-sharing-based sealed-bid e-auction schemes is provided in Table 2. [3] cannot achieve bid privacy and public verifiability at the same time. It is assumed in Table 2 that public verifiability, a more important property is achieved while bid privacy is sacrificed.

5.2 Efficiency Analysis

An efficiency comparison of the secret-sharing-based sealed-bid e-auction schemes is provided in Table 3. In Table 3, full-length exponentiations are counted, where Paillier encryption and RSA signature are assumed to be employed. It is illustrated in the two tables that compared to the previous secret-sharing-based sealed-bid e-auction schemes, the new scheme does not compromise efficiency while achieving much better properties. It is even more efficient than [10].

[2] A honest bidder can successfully settle a dispute as it is impossible to encrypt two different messages into the same ciphertext in Paillier encryption.

Table 3. Efficiency comparison

Auction schemes	bidder	auctioneer
[3]	3	$2n+1$
[4,6]	$2mw+1$	$1+2n+2\log_2 w$
[10]	$(2T+2+4m)w+1$	$1+6n+6\log_2 w$
New auction	$(T+3+2m)w+1$	$1+5n+5\log_2 w$

6 Conclusion

A new secret-sharing-based first-bid e-auction scheme is proposed. It can achieve all the desired properties for sealed-bid auctions at a reasonable cost. Moreover, attacks existing in the current secret-sharing-based sealed-bid e-auction schemes are prevented in the new scheme.

References

1. Feng Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *the Smart Card Research Conference, CARDIS'98*, volume 1820 of *Lecture Notes in Computer Science*, pages 213–220, Berlin, 1998. Springer-Verlag.
2. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Berlin, 1992. Springer-Verlag.
3. Matthew K Franklin and Michael K Reiter. The design and implementation of a secure auction service. In *IEEE Transactions on Software Enginerring*, volume 5, pages 302–312, May 1996.
4. H Kikuchi, Michael Harkavy, and J D Tygar. Multi-round anonymous auction. In *Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, pages 62–69, June 1998.
5. Hiroaki Kikuchi. (m+1)st-price auction. In *The Fifth International Conference on Financial Cryptography 2001*, volume 2339 of *Lecture Notes in Computer Science*, pages 291–298, Berlin, 2001. Springer-Verlag.
6. Hiroaki Kikuchi, Shinji Hotta, Kensuke Abe, and Shohachiro Nakanishi. Distributed auction servers resolving winner and winning bid without revealing privacy of bids. In *proc. of International Workshop on Next Generation Internet (NGITA2000), IEEE*, pages 307–312, July 2000.
7. P Paillier. Public key cryptosystem based on composite degree residuosity classes. In *EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Berlin, 1999. Springer-Verlag.
8. Torben P. Pedersen. Distributed provers with applications to undeniable signatures. In *EUROCRYPT '91*, pages 221–242, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science 547.
9. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *EUROCRYPT '91*, pages 129–140, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science 547.

10. Kun Peng, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Robust, privacy protecting and publicly verifiable sealed-bid auction. In *4th International Conference of Information and Communications Security, ICICS 2002*, volume 2513 of *Lecture Notes in Computer Science*, pages 147 – 159, Berlin, 2002. Springer.
11. C Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology, 4, 1991*, pages 161–174, 1991.
12. Adi Shamir. How to share a secret. *Communication of the ACM*, 22(11):612–613, Nov 1979.

Identity Based Delegation Network

Sherman S.M. Chow[*], Richard W.C. Lui, Lucas C.K. Hui, and S.M. Yiu

Department of Computer Science,
The University of Hong Kong,
Pokfulam, Hong Kong
{smchow, wclui, hui, smyiu}@cs.hku.hk

Abstract. Delegation of authorities is a common practice in various organizations. The way delegation is performed can be quite complicated. To capture possible delegation structures, the concept of *delegation network* is proposed, so that anyone can be convinced of who obtained delegation from whom in order to produce the final proxy signature. In this paper, we consider the delegation network for identity-based (ID-based) scenario. Since the public key is just a string denoting the user's identity, certificate management is simplified. Proxy signature schemes have been devised to delegate signing authorities. We show that a trivial attempt of extending an existing ID-based proxy signature may result in an insecure scheme. After that we propose a building block of our ID-based delegation network, which is an ID-based proxy signature supporting batch verifications. Our proposed ID-based delegation network is flexible in the sense that the whole delegation network does not need to be known in advance. Our proposal is provably secure under the random oracle model.

Keywords: Delegation network, identity-based cryptography, proxy signature, batch verification, bilinear pairings

1 Introduction

Delegation. Delegation is a process where a user (the delegator) grants some of his/her rights (power), e.g. the signing right, to another user (the delegate), to work on his/her behalf. It is a very common practice for users in an office to delegate their power to subordinates when they are on leave or need assistance. For a delegate to digitally sign on behalf of the delegator so that the receiver of the document be convinced that the signer has the signing right from the delegator, a straight-forward approach is to pass the delegator's signing key to the delegate. Obviously, this do not work well since the delegator has to change the key frequently. It also violates the non-repudiation requirement since it is difficult to prove who actually signed the document.

Proxy Signature. To tackle this problem, the notion of proxy signature was proposed in [21] to deal with the delegation of signing. In a proxy signature,

[*] Corresponding Author

the original signer creates a proxy key pair, denoted as (psk, ppk), using his/her own signing key (and possibly the delegate's public key). The delegate (called the proxy signer) signs a document using psk. The verifier has to use ppk as well as the public key of the original signer (again, and possibly the delegate's public key) to check the validity of the signature. Since the public key of the original signer is involved in the checking, the delegation relationship can be confirmed. Such schemes are very important technologies in various application domains, examples include but not limited to grid computing [9], distributed systems [22], distributed shared object systems [18] and a bunch of electronic commerce applications such as offline e-cash [23], privacy preserving signature in mobile communications [26], global distribution networks [2], and last but not least, mobile agents for electronic commerce.

Delegation Network. Hierarchical structure, which is common in organizations nowadays, complicates the way delegation is performed. Firstly, there can be chained delegation, in which the delegation occurs across more than two levels. For example, A may delegate her job to her subordinate B and B can further delegate the job to his subordinate C. Secondly, it is common that a signature is constructed by a group of members. In other words, the delegator can be a group of members instead of one single user. In this case, delegation can be passed from one group of users to another group of users. For example, A and B are required to sign together on a check. Now they both are on leave and so they may delegate the signing right to C and D.

To capture possible delegation structures, the concept of *delegation network* was proposed in [1]. The delegation structure of the signing group is modeled in a directed graph so that anyone can be convinced of who obtained delegation from whom in order to produce the final signature. An application of delegation network can be found in the use of mobile agents in electronic commerce application. Suppose mobile agents are ordered by a customer to search for a proper bid presented by a server and then digitally sign the server's bid together with the customer's requirement with both server's key and customer's key [16]. Consider a scenario that a mobile agent E is ordered to search for a travel package of lowest price (which includes both air ticket and hotel accommodation) offered by a travel agency on behalf of a research student. On the other hand, a mobile agent F is ordered by a travel agency to search for the prices of flight ticket and hotel accommodation. Then, agent E will delegate the signing authority to agent F, in which F will further delegate this signing authority to the airline company and the hotel, who sign their corresponding bid using the delegation received together with their respectively private key.

Identity-Based Cryptography. Most of the proxy signature schemes are based on a public key infrastructure (PKI). As an alternative to PKI, Shamir introduced the concept of identity-based (ID-based) signature schemes [27] and the design of ID-based schemes have attracted a lot of attention recently [3,5, 6,13,19,25,31,32,33]. For traditional PKI, the public key is a "random-looking" string that is unrelated to the user's identity, so a trusted-by-all party called certificate authority will issue certificate that cryptographically bind the public key string with the user's identity, in the form of a digital signature. In ID-based paradigm, the public key can be any string that can uniquely identify the user. A trusted-by-all party called private key generator (PKG) will authenticate the users and generates the private key corresponding to their identities on demand. The certificates management procedures are thus simplified. ID-based signature schemes have advantages over PKI-based counterpart since the certificate do not need to be retrieved and verified before the verification of signatures. A delegation network typically involves a number of delegators and signers, it is tedious to verify the digital certificate of each of them, which makes an ID-based delegation network highly desirable.

Previous Work. There are several related works, but they are either insecure, not general and flexible enough, or not ID-based.

The notion of *delegation network* was introduced in [1]; with a SPKI [8] based solution. In [5], an ID-based multi-proxy signature scheme was proposed, in which a single signer can delegate his/her signing authority to a group of proxy members, and the proxy signature can be generated only by the cooperation of all the signers in such a proxy group. It is obvious that [5] handles a very special case of a delegation network since the original signing entity is just one person and the delegation is of two levels only. Besides, interaction among the proxy signers is needed and prior knowledge on who are going to sign the messages are assumed to give the final *single* signature.

On the other hand, in a multi-proxy multi-signature scheme [14] or fully distributed proxy signature [11,12], the original signing entity is not confined to a single person but a set of members. An authorized subset of this signing group can delegate the signing right to another group of members. And this proxy group can generate a proxy signature on behalf of the original signing group only if an authorized subset of members cooperate. These schemes can be considered as a generalization of [5]. However, these schemes also handle a special case of a delegation network having two levels only. Moreover, [14] was shown to be insecure by [29], and [11,12] were shown to be insecure by [30].

In fact, there is another type of proxy signature schemes called threshold proxy signature [15], that requires t out of n proxy signers' cooperation to issue a valid proxy signature, while any $t-1$ proxy signers cannot. As pointed out by [5], multi-proxy signature scheme can be regarded as a special case of the (t, n) threshold proxy signature for $t = n$. After this work, other threshold proxy signature schemes were proposed [34,35]. However, their weaknesses were discussed in [17] and [10].

Contribution. We show that a trivial attempt of extending an existing ID-based proxy signature may result in an insecure scheme, what is lacking is an ID-based proxy signature supporting batch verifications. We propose an ID-based proxy signature supporting batch verifications and use it as a building block to construct an ID-based delegation network. In contrast with proxy multi-signature schemes or multi-proxy signature schemes, which possibly require cooperation among delegators or signers to perform delegation or sign the message respectively, our scheme supports autonomous delegation and signature generation (i.e. requires neither interaction among delegators or signers nor the prior knowledge on the whole delegation network).

2 Technical Preliminaries

2.1 Bilinear Pairings

Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \cdot)$ be two cyclic groups of prime order q. The bilinear pairing is given as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, which satisfy the following properties:

1. *Bilinearity*: For all $P, Q, R \in \mathbb{G}_1$, $\hat{e}(P+Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$, and $\hat{e}(P, Q+R) = \hat{e}(P, Q)\hat{e}(P, R)$.
2. *Non-degeneracy*: There exists $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1$.

Definition 1. *Given a generator P of a group $\mathbb{G}$ and a 2-tuple (aP, bP), the Computational Diffie-Hellman problem (CDH problem) is to compute abP.*

2.2 Review of an Existing ID-Based Proxy Signature

We review the first ID-based proxy signature scheme from bilinear pairings [33], which can be seen as an extension to the ID-based signature scheme in [13]. Note that we changed the verification slightly to cater for batch verifications.

– `Setup`: Let P be an arbitrary generator of $\mathbb{G}_1$, the PKG chooses $s \in \mathbb{F}_q^*$ randomly. It sets $P_{pub} = sP$. The master-key is s, which is kept secret and known only by itself. Let H and H_1 be two cryptographic hash functions where $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_1 : \{0,1\}^n \times \mathbb{G}_2 \rightarrow \mathbb{F}_q^*$. The system parameters are $\{\mathbb{G}_1, \mathbb{G}_2, q, n, P, P_{pub}, \hat{e}(\cdot,\cdot), H(\cdot), H_1(\cdot,\cdot)\}$.
– `KeyGen`: The user with identity $ID \in \{0,1\}^*$ submits ID to PKG. PKG sets the user's public key Q_{ID} to be $H(ID) \in \mathbb{G}_1$, computes the user's private key S_{ID} by $S_{ID} = sQ_{ID}$. Then PKG sends the private key to the user.
– `Delegate`: For a particular user ID_A with the secret key S_{ID_A}, to delegate his/her signing authority to ID_C, a message $m_{A,C} \in \{0,1\}^*$ stating the delegation condition will be created, then he/she follows the steps below.
  1. Choose $x_{A,C}$ from $\mathbb{F}_q^*$ randomly.
  2. Compute $r_{A,C} = \hat{e}(P, P)^{x_{A,C}}$.
  3. Compute $h_{A,C} = H_1(m_{A,C}, r_{A,C})$.
  4. Compute $U_{A,C} = h_{A,C}S_{ID_A} + x_{A,C}P$.
  5. Send $\{m_{A,C}, U_{A,C}, r_{A,C}\}$ to ID_C, where $U_{A,C}$ will be used for the generation of final proxy signature and $r_{A,C}$ certifies this delegation.
– `VerifyDelegation`: Now the delegate ID_C verifies the delegation token $\{m_{A,C}, U_{A,C}, r_{A,C}\}$ received from ID_A by checking whether $\hat{e}(P, U_{A,C}) = \hat{e}(P_{pub}, H_1(m_{A,C}, r_{A,C})Q_{ID_A}) \cdot r_{A,C}$ holds. If so, compute the proxy signing key $S_{A,C}$ by $S_{A,C} = H_1(m_{A,C}, r_{A,C})S_{ID_C} + U_{A,C}$.
– `ProxySign`: For the delegate ID_C with the secret key $S_{A,C}$ to sign a message $m \in \{0,1\}^*$, he/she follows the steps below.

1. Choose $x_{C,m}$ from $\mathbb{F}_q^*$ randomly.
2. Compute $r_{C,m} = \hat{e}(P,P)^{x_{C,m}}$.
3. Compute $h_{C,m} = H_1(m, r_{C,m})$.
4. Compute $U_{C,m} = h_{C,m}S_{A,C} + x_{C,m}P$.
5. Send $\{U_{C,m}, r_{C,m}, m_{A,C}, U_{A,C}, r_{A,C}\}$ to the verifier, where $\{U_{C,m}, r_{C,m}\}$ is the signature for message m.

– `VerifyProxySignature`: The verifier, who got the proxy signature in the form of $\{U_{C,m}, r_{C,m}, m_{A,C}, U_{A,C}, r_{A,C}\}$, accepts it is a valid proxy signature generated by ID_C on the message m after received the delegation from ID_A according to the warrant $m_{A,C}$, if the below equality holds:

$$\hat{e}(P, U_{C,m}) = [\hat{e}(P_{pub}, Q_{ID_A} + Q_{ID_C})^{H_1(m_{A,C}, r_{A,C})} \cdot r_{A,C}]^{H_1(m, r_{C,m})} \cdot r_{C,m}.$$

3 ID-Based Proxy Signature with Batch Verifications

We first consider the scenario that there are more than one delegator. It is well known that proxy signatures can be trivially obtained, by asking the delegator to issue a digital signature on the warrant stating the delegation condition to the delegates. Similarly, the situation for two delegators can be obtained by two such signatures. This give motivations for devising an ID-based proxy signature with batch verifications, such that the cost in verifying n ID-based proxy signature is less than n times the cost of verification of a single ID-based proxy signature.

3.1 Initial Attempt

We try to extend Zhang's scheme [33] to support multiple delegators. Consider the case for both of ID_A and ID_B to delegate the signing power to ID_C, the proxy signing key is given by: $S_{A+B,C} = H_1(m_{A,B}, r_{A,B})S_{ID_C} + U_{A,B} + H_1(m_{A,C}, r_{A,C})S_{ID_C} + U_{A,C}$, and the equality to check in the verification of the final proxy signature becomes:

$$\begin{aligned}\hat{e}(P, U_{C,m}) = {} & [\hat{e}(P_{pub}, Q_{ID_A} + Q_{ID_C})^{H_1(m_{A,C}, r_{A,C})} \cdot r_{A,C}]^{H_1(m, r_{C,m})} \\ & \cdot [\hat{e}(P_{pub}, Q_{ID_B} + Q_{ID_C})^{H_1(m_{B,C}, r_{B,C})} \cdot r_{B,C}]^{H_1(m, r_{C,m})} \cdot r_{C,m}.\end{aligned}$$

However, we found that it is not necessary to ask for the help of ID_B to generate such valid looking proxy signature, the attack is as follow.

1. Create the message m, the warrants $m_{A,C}$ and $m_{B,C}$ as desired.
2. Randomly choose R from $\mathbb{G}_1$.
3. Randomly choose $x_{B,C}, x_{C,m}$ from $\mathbb{F}_q^*$.
4. Compute $r_{B,C} = \hat{e}(P,P)^{x_{B,C}}$ and $r_{C,m} = \hat{e}(P,P)^{x_{C,m}}$.
5. Compute $r_{A,C} = \hat{e}(P,R)\hat{e}(P_{pub}, Q_{ID_B} + Q_{ID_C})^{-H_1(m_{B,C}, r_{B,C})}$.
6. Compute $U_{C,m} = H_1(m, r_{C,m}) \cdot H_1(m_{A,C}, r_{A,C})(S_{ID_A} + S_{ID_C}) + H_1(m, r_{C,m})$ $(R + x_{B,C}P) + x_{C,m}P$.

It is easy to see that S_{ID_B} is not involved in the above procedures and the proxy signature produced passes the verification algorithm.

$$\begin{aligned}
\hat{e}(P, U_{C,m}) &= \hat{e}(P, H_1(m, r_{C,m}) \cdot H_1(m_{A,C}, r_{A,C})(S_{ID_A} + S_{ID_C}) \\
&\quad + H_1(m, r_{C,m})(R + x_{B,C}P) + x_{C,m}P) \\
&= \hat{e}(P_{pub}, (Q_{ID_A} + Q_{ID_C}))^{H_1(m_{A,C}, r_{A,C})H_1(m, r_{C,m})} \hat{e}(P, R)^{H_1(m, r_{C,m})} \\
&\quad \hat{e}(P, P)^{x_{B,C} \cdot H_1(m, r_{C,m})} \hat{e}(P, P)^{x_{C,m}} \cdot r_{A,C}^{H_1(m, r_{C,m})} / r_{A,C}^{H_1(m, r_{C,m})} \\
&= [\hat{e}(P_{pub}, Q_{ID_A} + Q_{ID_C})^{H_1(m_{A,C}, r_{A,C})} \cdot r_{A,C}]^{H_1(m, r_{C,m})} r_{C,m} r_{B,C}^{H_1(m, r_{C,m})} \\
&\quad \cdot \hat{e}(P, R)^{H_1(m, r_{C,m})} / r_{A,C}^{H_1(m, r_{C,m})} \\
&= [\hat{e}(P_{pub}, Q_{ID_A} + Q_{ID_C})^{H_1(m_{A,C}, r_{A,C})} \cdot r_{A,C}]^{H_1(m, r_{C,m})} r_{C,m} \\
&\quad \cdot \hat{e}(P_{pub}, Q_{ID_B} + Q_{ID_C})^{H_1(m, r_{C,m}) \cdot H_1(m_{B,C}, r_{B,C})} r_{B,C}^{H_1(m, r_{C,m})} \\
&= [\hat{e}(P_{pub}, Q_{ID_A} + Q_{ID_C})^{H_1(m_{A,C}, r_{A,C})} \cdot r_{A,C}]^{H_1(m, r_{C,m})} \\
&\quad \cdot [\hat{e}(P_{pub}, Q_{ID_B} + Q_{ID_C})^{H_1(m_{B,C}, r_{B,C})} \cdot r_{B,C}]^{H_1(m, r_{C,m})} \cdot r_{C,m}.
\end{aligned}$$

In other words, the extended scheme is not unforgeable. Careful reader will find that the attack is relied on the fact that $r_{A,C}$ can be chosen according to the value of $r_{B,C}$. If the scheme requires both user ID_A and user ID_B to generates $r_{A+B,C}$ together, and generates the delegation key using the value $h_{A+B,C} = H_1(m_{A+B,C}, r_{A+B,C})$ then the above attack is not possible. However, this requires the whole delegation network to be known in advance.

3.2 A New ID-Based Signature Supporting Batch Verifications

We propose a new ID-based signature (IBS) scheme supporting batch verifications, which will be extended to support proxy signature later.

- `Setup`: Let P be an arbitrary generator of $\mathbb{G}_1$, the Private Key Generator (PKG) chooses $s \in \mathbb{F}_q^*$ randomly. It sets $P_{pub} = sP$. The master-key is s, which is kept secret and known only by itself. Let H and H_2 be two cryptographic hash functions where $H : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \{0,1\}^n \times \mathbb{G}_1 \to \mathbb{G}_1$. The system parameters are $\{\mathbb{G}_1, \mathbb{G}_2, q, n, P, P_{pub}, \hat{e}(\cdot,\cdot), H(\cdot), H_2(\cdot,\cdot)\}$.
- `KeyGen`: The user with identity $ID \in \{0,1\}^*$ submits ID to PKG. PKG sets the user's public key Q_{ID} to be $H(ID) \in \mathbb{G}_1$, computes the user's private key S_{ID} by $S_{ID} = sQ_{ID}$. Then PKG sends the private key to the user.
- `Sign`: For user ID to generate a signature on the message m, x is randomly chosen from $\mathbb{F}_q^*$, then the signature will be $\{R = xP, U = S_{ID} + xH_2(m, R)\}$.
- `BatchVerify`: The verifier who got the signatures $\{R_i, U_i\}$ from ID_i (or an aggregation of signatures $U = \sum U_i$, together with $\bigcup\{R_i\}$) accepts all of them are valid signatures on respective message m_i if $\hat{e}(P, U) = \hat{e}(P_{pub}, \sum Q_{ID_i}) \prod \hat{e}(R_i, H_2(m_i, R_i))$ holds.

  It is easy to see the signatures generated pass the verification algorithm.

$$\begin{aligned}\hat{e}(P, \sum U_i) &= \hat{e}(P, \sum (S_{ID_i} + x_i H_2(m_i, R_i))) \\ &= \hat{e}(P, \sum S_{ID_i})\hat{e}(P, \sum x_i H_2(m_i, R_i)) \\ &= \hat{e}(P_{pub}, \sum Q_{ID_i}) \prod \hat{e}(x_i P, H_2(m_i, R_i)) \\ &= \hat{e}(P_{pub}, \sum Q_{ID_i}) \prod \hat{e}(R_i, H_2(m_i, R_i))\end{aligned}$$

3.3 Analysis

The comparison of our scheme with the existing batch verifications scheme [32] is summarized in Table 1. Note that most previous ID-based signature schemes [3,6,13,19,25] take at least $2n$ pairing operations in verification of n signatures.

Table 1. Comparison on the proposed scheme and the existing scheme

	Tight Reduction [1]	Proxy Extension [2]	`Sign` (Single Message)			`BatchVerify` (n Signatures)		
Schemes			G_1 +	$G_1 \times$	$\hat{e}(\cdot,\cdot)$	G_1 +	$G_1 \times$	$\hat{e}(\cdot,\cdot)$
Proposed Scheme	✓	✓	1	2	0	n	0	$n+2$
Scheme in [32]	×	×	1	3	0	n	n	$n+1$

The above scheme can be seen as extended from the modified version of Sakai-Ogishi-Kasahara ID-based signature [25] in [3], in which the single verification version is proven to be existentially unforgeable against adaptive chosen-message-and-identity attack under CDH assumption or one-more Diffie-Hellman assumption [19]. Hence we only give the security analysis of our batch verifications against k-aggregate forger [32], which is defined as follows. Similar to [19] and [6], our reduction does not make use of forking lemma [24] which the scheme in [32] relied, so our scheme obtained a reduction with a tighter security bound.

Definition 2. *A k-aggregate forger is defined as a forger who produces a set of k signatures $(ID_1, ID_2, \cdots, ID_k, m_1, m_2, \cdots, m_k, \sigma_1, \sigma_2, \cdots, \sigma_k)$ which pass the batch verifications algorithm, and there exists $i \in \{1, 2, \cdots, k\}$ in which the pair (ID_i, m_i) has not been presented to the signing oracle, i.e. σ_i does not comes from the signing oracle.*

Theorem 1. *If there is a k-aggregate forger $\mathcal{F}$ which succeeds in producing a set of k signatures with probability ϵ by launching an adaptive chosen-message-and-identity attack, making at most q_H queries for hashing the identities, q_{H_2} queries for hashing the messages, q_E key generation queries and q_S signing queries, then the CDHP can be solved with an advantage $\epsilon' > \frac{\epsilon-(q_S(q_{H_2}+q_S)+1)/2^l}{e(q_E)}$.*

[1] Whether a tight security bound with the underlying hard problem can be obtained.
[2] Whether there exists provably secure extension supporting proxy signing.

Proof. Suppose that there exists a k-aggregate forger $\mathcal{F}$ that has advantage ϵ in attacking our IBS. We show that an algorithm $\mathcal{C}$ can be constructed to solve the CDHP in $\mathbb{G}_1$. That is, given (P, aP, bP), algorithm $\mathcal{C}$ is able to compute abP, assuming $\mathcal{F}$ will ask for $H(ID)$ before ID is used in any other query. During the game, $\mathcal{F}$ will consult $\mathcal{C}$ for answers to the random oracles H and H_2, $\mathcal{C}$ will keep lists L_1 and L_2 to store the answers used respectively. Let $P_{pub} = bP$. Algorithm $\mathcal{C}$ interacts with $\mathcal{F}$ in the *existential unforgeability against adaptive chosen-message-and-identity attack game* as follows.

Queries on oracle H for identity: We embed part of the challenge aP in the answer of many H queries [7]. When $\mathcal{F}$ asks queries on the hash value of identity ID, $\mathcal{C}$ picks $y_1 \in_R \mathbb{F}_q^*$ and repeats the process until y_1 is not in the list L_1. $\mathcal{C}$ then flips a coin $W \in \{0, 1\}$ that yields 0 with probability ζ and 1 with probability $1 - \zeta$. (ζ will be determined later.) If $W = 0$ then the hash value $H(ID)$ is defined as $y_1 P$; else if $W = 1$ then returns $H(ID) = y_1(aP)$. In either case, $\mathcal{C}$ stores (ID, y_1, W) in the list L_1.

Queries on oracle H_2: When $\mathcal{F}$ asks queries on the hash value of (m, R), $\mathcal{C}$ returns the hash value from the list L_2 is it is already defined; else $\mathcal{C}$ picks $y_2 \in_R \mathbb{F}_q^*$ and repeats the process until y_2 is not in the list L_2. $\mathcal{C}$ returns $y_2 P$ and stores (m, R, y_2) in the list L_2.

Private key generation queries: When $\mathcal{F}$ asks for the private key of user ID, $\mathcal{C}$ looks for (ID, y_1, W) entry in L_1. If $W = 1$ then $\mathcal{C}$ outputs "failure" and halts since $\mathcal{C}$ does not know how to compute $y_1 abP$. Otherwise a valid private key $y_1(bP)$ is returned.

Sign queries: When $\mathcal{F}$ asks for a signature of user ID on message m, $\mathcal{C}$ looks for (ID, y_1, W) entry in L_1. If $W = 0$, no matter what the value $H_2(m)$ is, $\mathcal{C}$ output the signature using the `Sign` algorithm since $\mathcal{C}$ knows the corresponding private key. If $W = 1$, $\mathcal{C}$ picks $x_1, x_2 \in_R \mathbb{F}_q^*$, computes $U = x_1(bP)$ and $R = x_2(bP)$, and defines $H_2(m, R)$ as ${x_2}^{-1}(x_1 P - Q_{ID})$ (find another pair of (x_1, x_2) if collision occurs). The signature to be output is (U, R).

Forgery: Algorithm $\mathcal{F}$ returns a forgery in the form of a set of k-signatures $(m_1, m_2, \cdots, m_k, R_1, R_2, \cdots, R_k, U = U_1 + U_2 + \cdots + U_k)$, where (R_i, U_i) is the signature from the signer ID_i on the message m_i for $i \in \{1, 2, \cdots, k\}$.

Probability of success: $\mathcal{C}$ can solve the CDHP if there exists an $i \in \{1, 2, \cdots, k\}$ where (R_i, U_i) does not comes from the signing oracle and $(ID_i, y_1, 1)$ is in the list L_1. Otherwise $\mathcal{C}$ cannot compute abP from the forgery made by $\mathcal{F}$.

The probability that $\mathcal{C}$ answers to all private key generation queries is ζ^{q_E}, and the probability that $\mathcal{F}$ makes a forged signature for user ID_i where $(ID_i, y_1, 1)$ is in the list L_1 is $1 - \zeta$ (notice that other pair of (R_j, U_j) may come from the signing oracle). Hence the probability for all above events to occur is $f_{q_E}(\zeta)$ where $f_x(\zeta) = \zeta^x(1 - \zeta)$. Simple differentiation shows that $f_x(\zeta)$ is maximized when $\zeta = 1 - (x + 1)^{-1}$, and the maximized probability is $\frac{1}{x}(1 - \frac{1}{x+1})^{x+1}$, which is $\frac{1}{q_E}(1 - \frac{1}{q_E+1})^{q_E+1}$. For large q_E, this probability is equal to $1/eq_E$.

$\mathcal{C}$ may also fail if a signing query conflicts with a H_2 query, since L_2 contains at most $q_{H_2} + q_S$ entries, the probability of such conflict is at most $q_S(q_{H_2} + q_S)/2^l$. Besides, the rare case (the probability is at most $1/2^l$, where l is the security parameter) that $\mathcal{F}$ makes a valid forgery without asking for the value of $H_2(m_i, R_i)$ for $i \in \{1, 2, \cdots, k\}$ can also make $\mathcal{C}$ fails. Since $\mathcal{F}$ succeeds in making a forgery with probability ϵ, taking into accounts of all these probabilities, the probability for $\mathcal{C}$ to solve the CDHP successfully is at most $\frac{\epsilon-(q_S(q_{H_2}+q_S)+1)/2^l}{e(q_E)}$.

Solving CDHP: If $\mathcal{F}$ makes a forged signature on message m_i for user ID_i where $(ID_i, y_1, 1)$ is in the list L_1, CDHP can be solved, the argument is as follows.

$$\begin{aligned}
\hat{e}(P,U) &= \hat{e}(P_{pub}, \sum Q_{ID_j}) \prod \hat{e}(R_j, H_2(m_j, R_j)) \\
\hat{e}(P,U) &= \hat{e}(bP, \sum Q_{ID_j}) \prod \hat{e}(R_j, {y_2}_j P) \\
&\quad ({y_2}_j \text{ can be obtained by looking up the list} L_2) \\
\hat{e}(P,U) &= \hat{e}(bP, Q_{ID_i}) \hat{e}(bP, \sum_{j \neq i} Q_{ID_j}) \prod \hat{e}(P, {y_2}_j R_j) \\
\hat{e}(P,U) &= \hat{e}(bP, {y_1}_i(aP)) \hat{e}(bP, \sum_{j \neq i} {y_1}_j(P)) \hat{e}(bP, \sum_{k \neq i} {y_1}_k(aP)) \prod \hat{e}(P, {y_2}_j R_j) \\
&\quad ({y_1}_i \text{ and } {y_1}_k \text{ can be obtained by finding } (ID, {y_1}_i, 1), \\
&\quad {y_1}_j \text{ can be obtained by finding } (ID, {y_1}_j, 0) \text{ in the list} L_1) \\
\hat{e}(P,U) &= \hat{e}(P, {y_1}_i(abP)) \hat{e}(P, \sum_{j \neq i} {y_1}_j(bP)) \hat{e}(P, \sum_{k \neq i} {y_1}_k(abP)) \prod \hat{e}(P, {y_2}_j R_j) \\
U &= {y_1}_i(abP) + \sum_{j \neq i} {y_1}_j(bP) + \sum_{k \neq i} {y_1}_k(abP) + \sum {y_2}_j R_j (\text{from the bilinearity}) \\
abP &= ({y_1}_i + \sum_{k \neq i} {y_1}_k)^{-1}(U - \sum_{j \neq i} {y_1}_j(bP) - \sum {y_2}_j R_j)
\end{aligned}$$

□

3.4 Extension to ID-Based Proxy Signature

We can extend the above ID-based signature scheme supporting batch verifications to its proxy signature version using the same idea as Zhang's scheme [33]. We can first use the signing algorithm to sign on a message warrant, the delegate verify this delegation by using the signature verification algorithm. If it passes, then the delegate can generate the secret proxy signing key by using the signature and his/her own private key. A realization of this idea can be found in [31]. Due to space limitation, we do not describe this scheme explicitly, since it can be seen as a special case for our ID-based delegation network.

4 ID-Based Delegation Network

In this section we combine our ID-based proxy signature scheme supporting batch verifications with the provably secure ID-based proxy signature scheme proposed in [31] to devise a scheme supporting ID-based delegation network.

4.1 Notation

The delegation structure is defined similar to the signing structure in [20]. Let $\mathbb{ID} = \{ID_1, \cdots, ID_n\}$ be a group of co-signers. Define the delegation structure Λ for $\mathbb{ID}$ as a directed graph, which contain all u_i where $ID_i \in \mathbb{ID}$ as real nodes together with two dummy nodes: u_0, a dummy node denotes the starting node and u_∞, a dummy node denotes the terminal node. A directed edge pointing from u_i to u_j means ID_i delegates the signing authority to ID_j. Furthermore, we denote $prev(ID_i)$ as the set of nodes directly precede to u_i in Λ. An example of the directed graph modeling the delegation network can be found at Figure 1. We also use the below notations to denote the set of message warrants and "public proxy key" certifying each delegation: $\mathbb{M}_{D,C} = \mathbb{M}_D \bigcup \{m_{D,C}\}$, $\mathbb{M}_i = \bigcup_{ID_D \in prev(ID_i)}(\mathbb{M}_D)$, $\mathbb{R}_{D,C} = \mathbb{R}_D \bigcup \{R_{D,C}\}$, and $\mathbb{R}_i = \bigcup_{ID_D \in prev(ID_i)}(\mathbb{R}_D)$, where D and C refers to the delegator and the delegate respectively.

4.2 Construction

- `Setup`: Let P be an arbitrary generator of $\mathbb{G}_1$, the Private Key Generator (PKG) chooses $s \in \mathbb{F}_q^*$ randomly. It sets $P_{pub} = sP$. The master-key is s, which is kept secret and known only by itself. Let H, H_3 and H_4 be three cryptographic hash functions where $H : \{0,1\}^* \to \mathbb{G}_1$, $H_3 : \{0,1\}^* \times \mathbb{G}_2 \to \mathbb{G}_1$ and $H_4 : \{0,1\}^n \times \mathbb{G}_2 \to \mathbb{F}_q^*$. The system parameters are $\{\mathbb{G}_1, \mathbb{G}_2, q, n, P, P_{pub}, \hat{e}(\cdot,\cdot), H(\cdot), H_3(\cdot,\cdot), H_4(\cdot,\cdot)\}$.
- `KeyGen`: Same as the above scheme.
- `Delegate`: For a particular user ID_D with the *secret proxy signing key* PS_D, to delegate his/her signing authority to ID_C, a message $m_{D,C} \in \{0,1\}^*$ stating the delegation condition (with at least the delegation relationship (D,C)) will be created, then he/she follows the steps below.
  1. Randomly choose $x_{D,C}$ from $\mathbb{F}_q^*$.
  2. Compute $R_{D,C} = x_{D,C}P$.
  3. Compute $H_{D,C} = H_3(ID_D||m_{D,C}, R_{D,C})$.
  4. Compute $U_{D,C} = PS_D + x_{D,C}H_{D,C}$ (If ID_D received no delegation before, set PS_D to S_D).
  5. Send $\{U_{D,C}, \mathbb{R}_{D,C}, \mathbb{M}_{D,C}\}$ to ID_C, where $U_{D,C}$ will be used for the generation of final proxy signature and $R_{D,C}$ is the public proxy key certifying this delegation. We may also include n, the number of distinct delegators involved in delegating the signing right to ID_A. (e.g. $n = 5$ for user u_6 in Figure 1). Its purpose is for an easier reconstruction of the graph representing the delegation network.
- `VerifyDelegation`: Given the delegates' identities $\mathbb{ID}$, the warrants $\mathbb{M}_C$, the public proxy key $\mathbb{R}_C$, the verifier ID_C follows the steps below to verify each delegation he/she received from the D.
  1. Compute

$$V = \sum_{ID_i \in \mathbb{ID}} [numpath_{ID_C}(ID_i) \cdot (\sum_{R_{i,j} \in \mathbb{R}_{D,C}} H_4(ID_i||ID_j||m_{D,C}, R_{i,j}))Q_{ID_i}],$$

where

- $numpath_{ID_C}(ID_i)$ is defined as the number of the paths from the node ID_i to the node ID_c in the delegation network. For examples, $numpath_{ID_4}(ID_i) = 1$ for $i \in \{1, 2, 3\}$ and $numpath_{ID_6}(ID_2) = numpath_{ID_6}(ID_3) = 2$. Notice that *numpath* can be computed easily, e.g. by adjacency matrix multiplication.
- $R_{i,j} \in \mathbb{R}_{D,C}$ means there is an edge from the node ID_i to the node ID_j in the delegation network, e.g. $R_{i,j} \in \mathbb{R}_{4,6} = \{(1,4), (2,3), (3,4)\}$ and $R_{i,j} \in \mathbb{R}_{6,\infty} = \{(1,4), (1,5), (2,3), (3,4), (3,5), (4,6), (5,6)\}$.

2. Compute $\rho = \prod_{R_{i,j} \in \mathbb{R}_{D,C}} \hat{e}(R_{i,j}, H_3(ID_D || m_{i,j}, R_{i,j}))$.
3. Check whether $\hat{e}(P, U_{D,C}) = \hat{e}(P_{pub}, V)\rho$.

If so, compute the proxy signing key PS_C by

$$PS_C = \sum_{ID_D \in prev(ID_C)} [H_4(ID_D || ID_C || m_{D,C}, R_{D,C}) S_{ID_C} + U_{D,C}].$$

– `ProxySign`: For the delegate ID_C with the proxy signing key PS_C to sign a message $m_{C,\infty} \in \{0,1\}^*$, he/she follows the steps below.
  1. Choose $x_{C,\infty}$ from $\mathbb{F}_q^*$ randomly.
  2. Compute $R_{C,\infty} = x_{C,\infty} P$.
  3. Compute $H_{C,\infty} = H_3(ID_C || m_{C,\infty}, R_{C,\infty})$.
  4. Compute $U_{C,\infty} = PS_C + x_{C,\infty} H_{C,\infty}$.
  5. Send $\{U_{C,\infty}, \mathbb{R}_{C,\infty}, \mathbb{M}_{C,\infty}\}$ to an appointed secretary, who just combines the partial proxy signatures to generate the final proxy signature for the whole delegation structure.

  The appointed secretary combines the partial proxy signature to give the final one: $\{U = \sum_{ID_y \in prev(ID_\infty)} (U_{y,\infty}), \mathbb{R}_\infty, \mathbb{M}_\infty\}$.
– `VerifyProxySignature`: The verifier who got the final proxy signature follows the steps below to verify the delegation network involved and the final signatures of the delegates.
  1. Compute $V = \sum_{ID_i \in \mathbb{ID}} [numpath_{ID_\infty}(ID_i) \cdot X_i]$,
     where $X_i = (\sum_{R_{i,j}: j \neq \infty, R_{i,j} \in \mathbb{R}_\infty} H_4(ID_i || ID_j || m_{D,C}, R_{i,j})) Q_{ID_i}$.
  2. Compute $\rho = \prod_{R_{i,j} \in \mathbb{R}_\infty} \hat{e}(R_{i,j}, H_3(ID_i || m_{i,j}, R_{i,j}))$.
  3. Accept if $\hat{e}(P, U) = \hat{e}(P_{pub}, V)\rho$, reject otherwise.

The consistency analysis of the scheme is similar to that of the basic version. A difference is the multiple involvement of secret key in the delegation, which contributes to the term $numpath_{ID_\infty}(ID_i)$ and the summation of H_4 values in verification equation. This point will become more clear in the below example.

4.3 Examples

Consider how the proxy signature is generated from the delegation network depicted in Figure 1. It is easy to see what user ID_4 received from users ID_1 and ID_3, and also what user ID_5 has received. Now we consider how the proxy signature signed by this For the sake of brevity, we start by showing what users ID_4 and ID_5 will do, and omit the verification of the delegation.

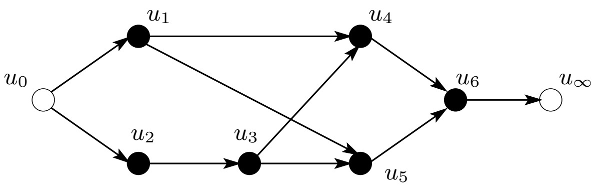


Fig. 1. Signing Structure

- User ID_4 does the following.
  1. User ID_4 receives from user ID_1: $(U_{1,4} = S_1 + x_{1,4}H_3(ID_1||m_{1,4}, x_{1,4}P), \mathbb{M}_{1,4} = \{m_{1,4}\}, \mathbb{R}_{1,4} = \{R_{1,4} = x_{1,4}P\})$.
  2. User ID_4 receives from user ID_3: $(U_{3,4} = S_2 + x_{2,3}H_3(ID_2||m_{2,3}, R_{2,3}) + H_4(ID_2, ID_3, m_{2,3}, R_{2,3})S_3 + x_{3,4}H_3(ID_3||m_{3,4}, R_{3,4})$, $\mathbb{M}_{3,4} = \{m_{2,3}, m_{3,4}\}, \mathbb{R}_{3,4} = \{R_{2,3}, R_{3,4}\})$.
  3. After verification of these delegations, compute the proxy signing key:
  $$PS_4 = H_4(ID_1||ID_4||m_{1,4}, R_{1,4})S_4 + U_{1,4} + H_4(ID_3||ID_4||m_{3,4}, R_{3,4})S_4 + U_{3,4}.$$
  4. Choose $x_{4,6}$ from $\mathbb{F}_q^*$ randomly.
  5. Create a warrant $m_{4,6}$.
  6. Compute $R_{4,6} = x_{4,6}P$.
  7. Compute $H_{4,6} = H_3(ID_4||m_{4,6}, R_{4,6})$.
  8. Compute $U_{4,6} = PS_4 + x_{4,6}H_{4,6}$.
  9. Send $(U_{4,6}, \mathbb{M}_{4,6} = \{m_{1,4}, m_{2,3}, m_{3,4}, m_{4,6}\}, \mathbb{R}_{4,6} = \{R_{1,4}, R_{2,3}, R_{3,4}, R_{4,6}\})$ to ID_6.
- User ID_5 performs a similar computation as that of ID_4, and sends $(U_{5,6}, \mathbb{M}_{5,6} = \{m_{1,5}, m_{2,3}, m_{3,5}, m_{5,6}\}, \mathbb{R}_{5,6} = \{R_{1,5}, R_{2,3}, R_{3,5}, R_{5,6}\})$ to ID_6, where $U_{5,6} = H_4(ID_1||ID_5||m_{1,5}, R_{1,5})S_5 + U_{1,5} + H_4(ID_3||ID_5||m_{3,5}, R_{3,5})S_5 + U_{3,5} + x_{5,6}H_{5,6}$.
- User ID_6 does the following to sign on the message $m_{6,\infty}$.
  1. After verifying these delegations, compute the proxy signing key: $PS_6 = H_4(ID_4||ID_6||m_{4,6}, R_{4,6})S_6 + U_{4,6} + H_4(ID_5||ID_6||m_{5,6}, R_{5,6})S_6 + U_{5,6}$.
  2. Choose $x_{6,\infty}$ from $\mathbb{F}_q^*$ randomly.
  3. Compute $R_{6,\infty} = x_{6,\infty}P$.
  4. Compute $H_{6,\infty} = H_3(ID_4||m_{6,\infty}, R_{6,\infty})$.
  5. Compute $U_{6,\infty} = PS_6 + x_{6,\infty}H_{4,6}$.
  6. Send $(U_{6,\infty}$, $\mathbb{M}_{6,\infty} = \{m_{1,4}, m_{1,5}, m_{2,3}, m_{3,4}, m_{3,5}, m_{4,6}, m_{5,6}\}$, $\mathbb{R}_{6,\infty} = \{R_{1,4}, R_{1,5}, R_{2,3}, R_{3,4}, R_{3,5}, R_{4,6}, R_{5,6}\})$ to an arbitrary secretary .
- The final proxy signature produced by the secretary is $(U, \mathbb{M}_\infty, \mathbb{R}_\infty)$, where
  1. $U = U_{6,\infty}$,
  2. $\mathbb{M}_\infty = \{m_{1,4}, m_{1,5}, m_{2,3}, m_{3,4}, m_{3,5}, m_{4,6}, m_{5,6}, m\}$ and
  3. $\mathbb{R}_\infty = \{R_{1,4}, R_{1,5}, R_{2,3}, R_{3,4}, R_{3,5}, R_{4,6}, R_{5,6}, R_{6,\infty}\}$.

– The verifier who got the final proxy signature follows the steps below to verify the delegation network involved and the final signatures of the delegates.

  1. Compute $V = \sum_{ID_i \in \mathbb{ID}} [numpath_{ID_\infty}(ID_i) \cdot X_i]$,
     where $X_i = (\sum_{R_{i,j}: j \neq \infty, R_{i,j} \in \mathbb{R}_\infty} H_4(ID_i || ID_j || m_{D,C}, R_{i,j}))Q_{ID_i}$.
     In this example, $V = 2Q_{ID_1} + 2Q_{ID_2} + 2H_4(ID_2||ID_3||m_{2,3}||R_{2,3})Q_{ID_3}$
     $+(H_4(ID_1||ID_4||m_{1,4}||R_{1,4}) + H_4(ID_3||ID_4||m_{3,4}||R_{3,4}))Q_{ID_4}$
     $+(H_4(ID_1||ID_5||m_{1,5}||R_{1,5}) + H_4(ID_3||ID_5||m_{3,5}||R_{3,5}))Q_{ID_5}$
     $+(H_4(ID_4||ID_6||m_{4,6}||R_{4,6}) + H_4(ID_5||ID_6||m_{5,6}||R_{5,6}))Q_{ID_6}$.
  2. Compute $\rho = \prod_{R_{i,j} \in \mathbb{R}_\infty} \hat{e}(R_{i,j}, H_3(ID_i||m_{i,j}, R_{i,j}))$, which is
     $\hat{e}(R_{1,4}, H_3(ID_1||m_{1,4}, R_{1,4})) \cdot \hat{e}(R_{1,5}, H_3(ID_1||m_{1,5}, R_{1,5}))$
     $\cdot\hat{e}(R_{2,3}, H_3(ID_2||m_{2,3}, R_{2,3}))\ \cdot\hat{e}(R_{3,4}, H_3(ID_3||m_{3,4}, R_{3,4}))$
     $\cdot\hat{e}(R_{3,5}, H_3(ID_3||m_{3,5}, R_{3,5}))\ \cdot\hat{e}(R_{4,6}, H_3(ID_4||m_{4,6}, R_{4,6}))$
     $\cdot\hat{e}(R_{5,6}, H_3(ID_5||m_{5,6}, R_{5,6}))\ \cdot\hat{e}(R_{6,\infty}, H_3(ID_6||m_{6,\infty}, R_{6,\infty}))$.
  3. Accept if $\hat{e}(P, U) = \hat{e}(P_{pub}, V)\rho$, reject otherwise.

4.4 Analysis

We first analyze the computational efficiency of the scheme. Although the formula looks complicated, the scheme is not inefficient. The delegation and signing requires no pairing computation at all. The verification process takes N scalar-point multiplication and E pairing operations, where N and E is the number of nodes and the number of edges in the graph representing the delegation network respectively. Notice that our scheme can be further optimized since the multiplication of a series of pairings in `VerifyProxySignature` can be optimized by using the concept of "Miller lite" of Tate pairing presented in [28]. The size of the signature is also reasonable. Without considering the warrants involved, our signature is of size $E + 1$ $\mathbb{G}_1$ elements, where each $\mathbb{G}_1$ element is about 300 bits only, for 923-bit discrete logarithm security. This analysis shows that our scheme is obviously more efficient than the trivial solution of issuing E digital signatures certifying the delegation conditions to the delegates or E invocations of existing ID-based proxy signature scheme.

For the security of our scheme, we consider the same model as the security model of proxy signature in [31] which models the extreme case that the adversary is working against a single honest user (says ID_1) and can get the private keys of all other users. We found that if there is an efficient forger that can make a forged signature or forged proxy signature, then we can use it to break our ID-based signature scheme supporting batch verifications.

Full proof of security is omitted due to space limitation. Now we highlight some of the key ideas. In our proof of security, the simulation of private key generation, delegation, standard signing and the proxy signing requests can be done in more or less the same way as those in the proof of [31]. The adversary is considered to be a successful forger of our ID-based delegation network if either one of the following forgeries are produced:

1. A standard signature by user ID_1 for a message that was not submitted to the standard signing oracle.

2. A proxy signature from a delegation network, which involves ID_1 as one of the delegators or as one of the proxy signers.

If forgery of type (1) is produced, it implies that our ID-based signature scheme supporting batch verifications is forgeable. If type (2) forgery is produced, with non-negligible probability (since we are considering the extreme case that the adversary is given the private keys of all users except one) we can generate a multi-signature signed by user ID_1; by "cancelling" the "participations" (delegation or signing) of other users by deducting the related terms from final proxy signature U. Since we have proven the security of our underlying scheme, by contradiction, our delegation network is unforgeable. Indeed, our simulation can embed an instance of the CDHP to the identity of user ID_1 and the system parameter so that the either type of forgery can help us to solve the CDHP. The cancellation to get the ID_1's private key or the way to solve the CDHP are similar to the steps of solving the CDHP in the proof of Theorem 1.

4.5 Extra Features

Since our proposed ID-based delegation network is built on a signature scheme supporting batch verifications, the delegator does not need to know who are also delegating his/her signer power to the same delegate in advance. Similarly, to give a proxy signature, the proxy signer does not need to know who else are going to sign the same message too. Our scheme also supports multi-signature: the proxy signers can sign on different messages but we can verify the signatures on different messages in the same verification process.

5 Conclusion

We presented an ID-based proxy signature scheme, which allows efficient delegation chains through batch verification. The scheme is based on a new ID-based signature scheme with batch verification, which is of interest in itself and whose security is proven in the paper. Our proposal for ID-based delegation network is an improvement over the alternative of applying existing proxy signatures to each delegate in the network, which requires interaction among the proxy signers and prior knowledge on who are going to sign the messages.

Future research direction is to further minimize the storage or bandwidth requirement for the delegation scenario. For example, devising an ID-based aggregate signatures scheme [4], such that the signature certifying the delegation and the signature on message can be aggregated into a small signature.

References

1. Tuomas Aura. On the Structure of Delegation Networks. In *PCSFW: Proceedings of the Eleventh Computer Security Foundations Workshop*. IEEE Computer Society Press, 1998.

2. Arno Bakker, Maarten van Steen, and Andrew S. Tanenbaum. A Law-Abiding Peer-to-Peer Network for Free-Software Distribution. In *IEEE International Symposium on Network Computing and Applications (NCA'01), Cambridge, MA, October 8-10, 2001, Proceedings*, 2001.
3. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security Proofs for Identity-Based Identification and Signature Schemes. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286. Springer, 2004.
4. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer.
5. Xiaofeng Chen, Fangguo Zhang, and Kwangjo Kim. ID-based Multi-Proxy Signature and Blind Multisignature from Bilinear Pairings. In *KIISC conference*, 2003.
6. Sherman S.M. Chow. Verifiable Pairing and Its Applications. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications: 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, Revised Selected Papers*, volume 3325 of *Lecture Notes in Computer Science*, pages 170–187. Springer.
7. Jean-Sébastien Coron. On the Exact Security of Full Domain Hash. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000.
8. Carl M. Ellison, Bill Franz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI Certificate Theory, Simple Public Key Certificate, SPKI Examples. Internet draft, SPKI Working Group, Internet Engineering Task Force, September 1999.
9. Ian Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A Security Architecture for Computational Grids. In *Proceedings of the Fifth ACM conference on Computer and Communications Security*, pages 83–92. ACM Press, 1998.
10. Hossein Ghodosi and Josef Pieprzyk. Repudiation of Cheating and Non-Repudiation of Zhang's Proxy Signature Schemes. In Josef Pieprzyk, Reihaneh Safavi-Naini, and Jennifer Seberry, editors, *Information Security and Privacy, Fourth Australasian Conference, ACISP 1999, Wollongong, NSW, Australia, April 7-9, 1999, Proceedings*, volume 1587 of *Lecture Notes in Computer Science.* Springer, 1999.
11. Javier Herranz and Germán Sáez. Verifiable Secret Sharing for General Access Structures, with Application to Fully Distributed Proxy Signatures. In Rebecca N. Wright, editor, *Financial Cryptography, 7th International Conference, FC 2003, Guadeloupe, French West Indies, January 27-30, 2003, Revised Papers*, volume 2742 of *Lecture Notes in Computer Science*, pages 286–302, 2003.
12. Javier Herranz and Germán Sáez. Revisiting Fully Distributed Proxy Signature Schemes. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 356–370. Springer, 2004. Also available at Cryptology ePrint Archive, Report 2003/197.

13. Florian Hess. Efficient Identity Based Signature Schemes based on Pairings. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2003.
14. Shin-Jia Hwang and Chiu-Chin Chen. New Multi-Proxy Multi-Signature Schemes. *Applied Mathematics and Computation*, 147(1):57–67, 2004.
15. Seungjoo Kim, Sangjoon Park, and Dongho Won. Proxy Signatures, Revisited. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Information and Communication Security, First International Conference, ICICS'97, Beijing, China, November 11-14, 1997, Proceedings*, volume 1334 of *Lecture Notes in Computer Science*. Springer, 1997.
16. Byoungcheon Lee, Heesun Kim, and Kwangjo Kim. Secure Mobile Agent Using Strong Non-Designated Proxy Signature. In Vijay Varadharajan and Yi Mu, editors, *Information Security and Privacy, Sixth Australasian Conference, ACISP 2001, Sydney, Australia, July 11-13, 2001, Proceedings*, volume 2119 of *Lecture Notes in Computer Science*. Springer, 2001.
17. Narn-Yih Lee, Tzonelih Hwang, and Chih-Hung Wang. On Zhang's Nonrepudiable Proxy Signature Schemes. In Colin Boyd and Ed Dawson, editors, *Information Security and Privacy, Third Australasian Conference, ACISP 1998, Brisbane, Queensland, Australia, July 1998, Proceedings*, volume 1438 of *Lecture Notes in Computer Science*. Springer, 1998.
18. J. Leiwo, C. Hanle, P. Homburg, and A.S. Tanenbaum. Disallowing Unauthorized State Changes of Distributed Shared Objects. In Sihan Qing and Jan H.P. Eloff, editors, *Information Security for Global Information Infrastructures*. Kluwer Academic Publishers, 2000.
19. Benoît Libert and Jean-Jacques Quisquater. The Exact Security of an Identity Based Signature and its Applications. Cryptology ePrint Archive, Report 2004/102, 2004. Available at http://eprint.iacr.org.
20. Chih-Yin Lin, Tzong-Chen Wu, and Fangguo Zhang. A Structured Multisignature Scheme from the Gap Diffie-Hellman Group. Cryptology ePrint Archive, Report 2003/090, 2003. Available at http://eprint.iacr.org.
21. Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy Signature: Delegation of the Power to Sign Messages. In *IEICE Trans. Fundamentals*, volume E79-A, 9, Sep 1996.
22. B. Clifford Neuman. Proxy-Based Authorization and Accounting for Distributed Systems. In *Thirteenth International Conference on Distributed Computing Systems*, pages 283–291, 1993.
23. Takeshi Okamoto, Mitsuru Tada, and Eiji Okamoto. Extended Proxy Signatures for Smart Cards. In Masahiro Mambo and Yuliang Zheng, editors, *Information Security, Second International Workshop, ISW'99, Kuala Lumpur, Malaysia, November 1999, Proceedings*, volume 1729 of *Lecture Notes in Computer Science*. Springer, 1999.
24. David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, 13(3):361–396, 2000.
25. Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on Pairing over Elliptic Curve. In *Proceedings of Symposium on Cryptography and Information Security (SCIS 2000) C-20*, 2000.

26. Seung-Hyun Seo and Sang-Ho Lee. New Nominative Proxy Signature Scheme for Mobile Communication. In *Conference on Security and Protection of Information 2003, Brno, Czech Republic, April 28-30, 2003*, pages 149–154, 2003.
27. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO 1984, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985.
28. Jerome A. Solinas. ID-based Digital Signature Algorithms. Slide Show presented at 7th Workshop on Elliptic Curve Cryptography (ECC 2003), August 2003.
29. Hung-Min Sun, Bin-Tsan Hsieh, and C.T. Lin. Cryptanalysis of A New Multi-Proxy Multi-Signature Scheme. In *Twelfth National Information Security Conference (ISC 2002)*, 2002.
30. Guilin Wang, Feng Bao, Jianying Zhou, and Robert H. Deng. Security Analysis of Some Proxy Signatures. In Jong In Lim and Dong Hoon Lee, editors, *Information Security and Cryptology - ICISC 2003, 6th International Conference, Seoul, Korea, November 27-28, 2003, Revised Papers*, volume 2971 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2004. Also available at Cryptology ePrint Archive, Report 2003/196.
31. Jing Xu, Zhenfeng Zhang, and Dengguo Feng. ID-Based Proxy Signature Using Bilinear Pairings. Cryptology ePrint Archive, Report 2004/206, 2004. Available at http://eprint.iacr.org.
32. HyoJin Yoon, Jung Hee Cheon, and Yongdae Kim. Batch Verifications with ID-Based Signatures. In Choonsik Park and Seongtaek Chee, editors, *Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, volume 3506 of *Lecture Notes in Computer Science*, pages 233–248. Springer, 2005.
33. Fangguo Zhang and Kwangjo Kim. Efficient ID-Based Blind Signature and Proxy Signature from Bilinear Pairings. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *Information Security and Privacy, Eighth Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003, Proceedings*, volume 2727 of *Lecture Notes in Computer Science*, pages 312–323. Springer, 2003.
34. Kan Zhang. Nonrepudiable Proxy Signature Schemes. Available at http://citeseer.nj.nec.com/zhang97nonrepudiable.html.
35. Kan Zhang. Threshold Proxy Signature Schemes. In *Proceedings of the First International Information Security Workshop*, pages 282–290, 1997.

On Session Key Construction in Provably-Secure Key Establishment Protocols[⋆]

Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock

Information Security Institute,
Queensland University of Technology,
GPO Box 2434, Brisbane, QLD 4001, Australia
{k.choo, c.boyd, y.hitchcock}@qut.edu.au

Abstract. We examine the role of session key construction in provably-secure key establishment protocols. We revisit an ID-based key establishment protocol due to Chen & Kudla (2003) and an ID-based protocol 2P-IDAKA due to McCullagh & Barreto (2005). Both protocols carry proofs of security in a weaker variant of the Bellare & Rogaway (1993) model where the adversary is not allowed to make any Reveal query. We advocate the importance of such a (Reveal) query as it captures the known-key security requirement. We then demonstrate that a small change to the way that session keys are constructed in both protocols results in these protocols being secure without restricting the adversary from asking the Reveal queries in most situations. We point out some errors in the existing proof for protocol 2P-IDAKA, and provide proof sketches for the improved Chen & Kudla's protocol. We conclude with a brief discussion on ways to construct session keys in key establishment protocols.

1 Introduction

Key establishment protocols are used for distributing shared keying material in a secure manner. For example, today's cryptosystems, such as AES, use key establishment schemes to establish shared keying material. However, despite their importance, the difficulties of obtaining a high level of assurance in the security of almost any new, or even existing, protocols are well illustrated with examples of errors found in many such protocols years after they were published [1,12,20].

The treatment of computational complexity analysis adopts a deductive reasoning process whereby the emphasis is placed on a proven reduction from the problem of breaking the protocol to another problem believed to be hard. Such an approach for key establishment protocols was made popular by Bellare & Rogaway [4] who provided the first formal definition for a model of adversary capabilities with an associated definition of security (which we refer to as the BR93 model in this paper). Since then, many research efforts have been oriented

⋆ This work was partially funded by the Australian Research Council Discovery Project Grant DP0345775.

towards this end which have resulted in numerous protocols with accompanying computational proofs of security proposed in the literature. In 1995, Bellare and Rogaway analysed a three-party server-based key distribution (3PKD) protocol [5] using an extension to the BR93 model. A more recent revision to the BR93 model was proposed in 2000 by Bellare, Pointcheval and Rogaway [3]. In independent yet related work, Bellare, Canetti, & Krawczyk [2] built on the BR93 model and introduced a modular proof model. However, some drawbacks with this formulation were discovered and this modular proof model was subsequently modified by Canetti & Krawczyk [9], and will be referred to as the CK2001 model in this paper.

The BR93 model is probably one of the most widely used proof models in the computational complexity approach for protocol analysis. In the model, the probabilistic polynomial-time (PPT) adversary controls all the communications that take place between parties via a pre-defined set of oracle queries, namely: Send, Reveal, and Corrupt. The Reveal query allows an adversary to expose session keys for uncorrupted parties, whilst the Corrupt query allows the adversary to corrupt any principal at will, and thereby learn the complete internal state of the corrupted principal. We observe that several protocols proven secure in the BR93 model restrict the adversary from asking the Reveal query. However, we argue that such a query is realistic in a real-world implementation as an adversary is often assumed to have the capability to acquire session keys. Such a (Reveal) query is essential as it allows us to model the scenario whereby each session key generated in one protocol round is independent and determines whether the particular session key will be exposed if other secret keys are compromised. In other words, the Reveal query captures the known-key security requirement in key establishment protocols, whereby a protocol should still achieve its goal in the face of a malicious adversary who has learned some other session keys [7,14]. In addition, omission of the Reveal query to the owner of the Test session in the proof model could also result in protocols vulnerable to reflection attacks being proven secure in such a model.

We revisit an ID-based key establishment protocol due to Chen & Kudla [10] and an ID-based protocol 2P-IDAKA due to McCullagh & Barreto [18]. Both protocols are role-symmetric and carry proofs of security in the BR93 model. However, the existing proofs of both protocols restrict the adversary from asking any Reveal query. Their arguments follow on from earlier work of Blake-Wilson, Johnson, & Menezes [6] who pointed out that it does not seem possible for role-symmetric protocols to be secure in the BR93 model if the Reveal query is allowed. In recent work, Jeong, Katz, & Lee [15] present two protocols $\mathcal{TS}1$ and $\mathcal{TS}2$, both with proofs of security in the BR93 model. This work contradicts the claim of Blake-Wilson *et al.* [6] as both protocols $\mathcal{TS}1$ and $\mathcal{TS}2$ are similar to the protocols analysed by Blake-Wilson *et al.* [6] in the BR93 model, but without restricting the adversary from asking the Reveal query.

We examine the existing arguments on the restriction of the Reveal query. We then demonstrate that by making a simple change to the construction of the session key (and not changing the protocol details), we are able to prove Chen &

Kudla's protocol secure in an intermediate variant of the BR93 model whereby the adversary, $\mathcal{A}$, is allowed to ask all the queries available in the model except asking Reveal queries to the sessions owned by the partner of the target Test session. Although we are unable to prove the improved protocol secure in the BR93 model without restricting $\mathcal{A}$ from asking the Reveal query due to some technicality, the improved protocol does not appear to be suffering from any insecurities even if we allow $\mathcal{A}$ to ask any Reveal queries to the perceived partner of the target Test session. Furthermore, by allowing $\mathcal{A}$ to ask Reveal queries directed at the owner of the Test session in our proof, effectively means that the improved Chen & Kudla's protocol is secure against reflection attacks. We reveal some errors in the existing proof of protocol 2P-IDAKA [18] as well as the observation that the proof is in a restricted BR93 model whereby $\mathcal{A}$ does not generate the input to the Test session, which is not a normal assumption in the Bellare–Rogaway models [3, 4, 5].

The Importance of Session Key Construction: We observe that there is neither a formal definition of session key construction in the proof models nor the existence of a rule of thumb on how session keys in key establishment protocols should be constructed. Our case studies illustrate that the way session keys are constructed can have an impact on the security of the protocol in the model. It appears that certain ways of constructing a session key may contribute to the security of a key establishment protocol.

Surprisingly, no one has pointed out the importance of session key construction despite its significance to the security of key establishment protocols. Of course, we do not claim that session keys constructed in our proposed fashion will necessarily result in a provably-secure protocol as the security of the protocol is based on many other factors, such as the underlying cryptographic primitives used. However, we do claim that having a sound construction of session keys will reduce the number of possible attacks on the key establishment protocol.

We regard the main contributions of this paper to be of three-fold significance:

1. demonstrating that the ID-based protocols of Chen & Kudla and McCullagh & Barreto can be proven secure in an intermediate BR93 model whereby the restriction of the Reveal query is only on the responder partner and the owner of the Test session respectively,
2. identifying the importance of session key constructions in key establishment protocols and contributing towards a better understanding of how to construct secure session keys in key establishment protocols, and
3. identifying errors in the existing proof of protocol 2P-IDAKA [18].

Section 2 provides an informal overview of the BR93 model. Section 3 revisits the Chen–Kudla ID-based key establishment protocol. We present the arguments of the existing proof on why the Reveal query is not allowed, and present an improved protocol. We then explain why the Reveal query cannot be answered if the adversary $\mathcal{A}$ ask any Reveal queries to the partner player of the target Test session. We conclude this section with a sketch of the proof for the improved protocol. Section 4 revisits the McCullagh–Barreto protocol 2P-IDAKA. Similarly to Section 3, we present the arguments of the existing proof on why the Reveal query is not allowed. We also identify some errors in the existing proof of the protocol. We then present an improved protocol. Section 5 presents our proposal on how session keys should be constructed. Section 6 presents the conclusions.

2 The BR93 Model

In this section, a brief overview of the BR93 model is provided primarily for the benefit of the reader in understanding the model [4].

2.1 Adversarial Powers

The adversary $\mathcal{A}$ is defined to be a probabilistic machine that is in control of all communications between parties by interacting with two sets, $\Pi^i_{U_1,U_2}$ and $\Psi^j_{U_1,U_2}$ of oracles ($\Pi^i_{U_1,U_2}$ is defined to be the i^{th} instantiation of a principal U_1 in a specific protocol run and U_2 is the principal with whom U_1 wishes to establish a secret key). The predefined oracle queries are as follows:

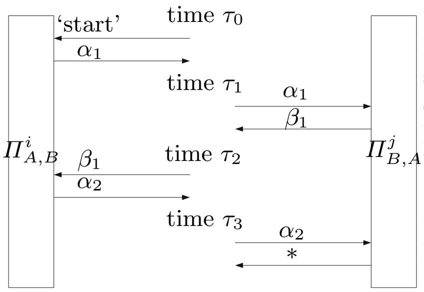
- Send(U_1, U_2, i, m) query computes a response according to the protocol specification and decision on whether to accept or reject yet, and returns them to $\mathcal{A}$.
- The client oracle, $\Pi^i_{U_1,U_2}$, upon receiving a Reveal(U_1, U_2, i) query, and if it has accepted and holds some session key, will send this session key back to $\mathcal{A}$.
- Corrupt(U_1, K_E) query allows $\mathcal{A}$ to corrupt the principal U_1 at will, and thereby learn the complete internal state of the corrupted principal. Note that such a query does not exist in the original BR93 model, but generally added by those using this model. In the Bellare & Rogaway (1995) model [5], the corrupt query also gives $\mathcal{A}$ the ability to overwrite the long-lived key of the corrupted principal with any value of her choice (i.e. K_E).
- Test(U_1, U_2, i) query is the only oracle query that does not correspond to any of $\mathcal{A}$'s abilities. If $\Pi^i_{U_1,U_2}$ has accepted with some session key and is being asked a Test(U_1, U_2, i) query, then depending on a randomly chosen bit b, $\mathcal{A}$ is given either the actual session key or a session key drawn randomly from the session key distribution.

2.2 Definition of Partnership

Partnership is defined using the notion of matching conversations, where a conversation is defined to be the sequence of messages sent and received by an oracle. The sequence of messages exchanged (i.e., only the Send oracle queries) are recorded in the transcript, T. At the end of a protocol run, T will contain the record of the Send queries and the responses as shown in Figure 1. Definition 1 gives a simplified definition of matching conversations for the case of the protocol shown in Figure 1.

Definition 1 (BR93 Definition of Matching Conversations [4]). *Let n be the maximum number of sessions between any two parties in the protocol run. Run the protocol shown in Figure 1 in the presence of a malicious adversary $\mathcal{A}$ and consider an initiator oracle $\Pi^i_{A,B}$ and a responder oracle $\Pi^j_{B,A}$ who engage in conversations C_A and C_B respectively. $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ are said to be partners if they both have matching conversations, where*

$$C_A = (\tau_0, 'start', \alpha_1), (\tau_2, \beta_1, \alpha_2)$$
$$C_B = (\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, *), \textit{ for } \tau_0 < \tau_1 < \dots$$



Note that the construction of conversation shown in Definition 1 depends on the number of parties and the number of message flows. Informally, both $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ are said to be BR93 partners if each one responded to a message that was sent unchanged by its partner with the exception of perhaps the first and last message.

Fig. 1. Matching conversation [4]

2.3 Definition of Freshness

The notion of freshness is used to identify the session keys about which $\mathcal{A}$ ought not to know anything because $\mathcal{A}$ has not revealed any oracles that have accepted the key and has not corrupted any principals knowing the key. Definition 2 describes freshness in the BR93 model, which depends on the notion of partnership in Definition 1.

Definition 2 (Definition of Freshness). *Oracle $\Pi^i_{A,B}$ is fresh (or it holds a fresh session key) at the end of execution, if, and only if, oracle $\Pi^i_{A,B}$ has accepted with or without a partner oracle $\Pi^j_{B,A}$, both oracle $\Pi^i_{A,B}$ and its partner oracle $\Pi^j_{B,A}$ (if such a partner oracle exists) have not been sent a* Reveal *query, and the principals A and B of oracles $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ (if such a partner exists) have not been sent a* Corrupt *query.*

2.4 Definition of Security

Security is defined using the game $\mathcal{G}$, played between a malicious adversary $\mathcal{A}$ and a collection of $\Pi^i_{U_x,U_y}$ oracles for players $U_x, U_y \in \{U_1, \dots, U_{N_p}\}$ and instances $i \in \{1, \dots, N_s\}$. The adversary $\mathcal{A}$ runs the game $\mathcal{G}$, whose setting is explained in Table 1.

Table 1. Setting of game $\mathcal{G}$

Stage 1:	$\mathcal{A}$ is able to send any oracle queries at will.
Stage 2:	At some point during $\mathcal{G}$, $\mathcal{A}$ will choose a fresh session on which to be tested and send a Test query to the fresh oracle associated with the test session. Depending on the randomly chosen bit b, $\mathcal{A}$ is given either the actual session key or a session key drawn randomly from the session key distribution.
Stage 3:	$\mathcal{A}$ continues making any oracle queries at will but cannot make Corrupt and/or Reveal that trivially expose the test session key.
Stage 4:	Eventually, $\mathcal{A}$ terminates the game simulation and outputs a bit b', which is its guess of the value of b.

Success of $\mathcal{A}$ in $\mathcal{G}$ is quantified in terms of $\mathcal{A}$'s advantage in distinguishing whether $\mathcal{A}$ receives the real key or a random value. $\mathcal{A}$ wins if, after asking a $\mathsf{Test}(U_1, U_2, i)$ query, where $\Pi^i_{U_1,U_2}$ is fresh and has accepted, $\mathcal{A}$'s guess bit b' equals the bit b selected during the $\mathsf{Test}(U_1, U_2, i)$ query. Let the advantage function of $\mathcal{A}$ be denoted by $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k})$, where $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k}) = 2 \times \Pr[\mathsf{b} = \mathsf{b}'] - 1$. Definition 3 describes security for the BR93 model.

Definition 3 (BR93 Definition of Security [4]**).** *A protocol is secure in the BR93 model if for all PPT adversaries $\mathcal{A}$,*

1. *if uncorrupted oracles $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ complete with matching conversations, then the probability that there exist i, j such that $\Pi^i_{A,B}$ accepted and there is no $\Pi^j_{B,A}$ that had engaged in a matching session is negligible.*
2. $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k})$ *is negligible.*

If both requirements of Definition 3 are satisfied, then $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ will also have the same session key.

3 Chen–Kudla ID-Based Protocol

Figure 2 describes protocol 2 of Chen & Kudla [10]. There are two entities in the protocols, namely initiator, A, and responder, B. The notation used in the protocols is as follows: $S_A = sQ_A$ and $S_B = sQ_B$ denote the private keys of A and B respectively, $\mathcal{H}$ denotes some secure hash function, $Q_A = \mathcal{H}(ID_A)$, $Q_A = \mathcal{H}(ID_B)$, $W_A = aQ_A$ and $W_B = bQ_B$ where W_A and W_B denote the ephemeral public keys of A and B respectively, and a and b are the ephemeral private keys of A and B respectively. At the end of the protocol execution, both A and B accept the session key $SK_{AB} = \mathcal{H}(\hat{e}(S_A, W_B + aQ_B))$ and $SK_{BA} = \mathcal{H}(\hat{e}(W_A + bQ_A, S_B))$ respectively.

3.1 Existing Arguments on the Restriction of Reveal Query

In the existing proof by Chen & Kudla [10, Proof of Theorem 1], they indicated that no Reveal query is allowed due to the description provided in Figure 3, where

$$\begin{array}{ccc}
A & & B \\
a \in_R \mathbb{Z}_q^* & \xrightarrow{W_A = aQ_A} & b \in_R \mathbb{Z}_q^* \\
K_{AB} = \hat{e}(S_A, W_B + aQ_B) & \xleftarrow{W_B = bQ_B} & K_{BA} = \hat{e}(W_A + bQ_A, S_B) \\
\multicolumn{3}{c}{K_{AB} = K_{BA} = \hat{e}(Q_A, Q_B)^{s(a+b)}} \\
\multicolumn{3}{c}{SK_{AB} = \mathcal{H}(K_{AB}) = SK_{BA} = \mathcal{H}(K_{BA})}
\end{array}$$

Fig. 2. Chen–Kudla Protocol 2

$$\begin{array}{ccccc}
A & & \mathcal{A} & & B \\
a \in_R \mathbb{Z}_q^* & \xrightarrow{W_A = aQ_A} & \text{Intercept} & & \\
 & & c \in_R \mathbb{Z}_q^* & \xrightarrow{W_A + cQ_A} & \\
 & \xleftarrow{W_B + cQ_B} & \text{Intercept} & \xleftarrow{W_B = bQ_B} & b \in_R \mathbb{Z}_q^* \\
\multicolumn{2}{c}{K_{AB} = \hat{e}(S_A, W_B + cQ_B + aQ_B)} & & \multicolumn{2}{c}{K_{BA} = \hat{e}(W_A + bQ_A + cQ_A, S_B)} \\
\multicolumn{5}{c}{K_{AB} = K_{BA} = \hat{e}(Q_A, Q_B)^{s(a+b+c)}} \\
\multicolumn{5}{c}{SK_{AB} = \mathcal{H}(K_{AB}) = SK_{BA} = \mathcal{H}(K_{BA})}
\end{array}$$

Fig. 3. Execution of Chen-Kudla protocol 2 in the presence of a malicious adversary

Figure 3 describes the execution of the protocol in the presence of a malicious adversary, $\mathcal{A}$.
At the end of the protocol execution, neither A nor B are partnered since they do not have matching conversations (as described in Definition 1 in Section 2), as A's transcript is $(W_A, W_B + cQ_B)$ whilst B's transcript is $(W_A + cQ_A, W_B)$. However, both A and B accept the same session key $K_{AB} = K_{BA} = \hat{e}(Q_A, Q_B)^{s(a+b+c)}$. Therefore, $\mathcal{A}$ is able to trivially expose a fresh session key by asking a Reveal query to a non-partner oracle. Therefore, the protocol will not be secure if $\mathcal{A}$ is allowed access to a Reveal query. Similar arguments apply for the remaining three protocols of Chen & Kudla [10].

3.2 Improved Chen–Kudla Protocol

Let A's transcript be denoted by $\mathcal{T}_A$ and B's transcript be denoted by $\mathcal{T}_B$. Consider the scenario whereby session keys of A and B (denoted as SK_{AB} and SK_{BA} respectively) are constructed as

$$\begin{aligned}
SK_{AB} = \mathcal{H}(K_{AB}) &= \mathcal{H}(A||B||\mathcal{T}_A||\hat{e}(S_A, W_B + aQ_B)) \\
&= \mathcal{H}(A||B||\mathcal{T}_A||\hat{e}(Q_A, Q_B)^{s(a+b)}), \\
SK_{BA} = \mathcal{H}(K_{BA}) &= \mathcal{H}(A||B||\mathcal{T}_B||\hat{e}(W_A + bQ_A, S_B) \\
&= \mathcal{H}(A||B||\mathcal{T}_B||\hat{e}(Q_A, Q_B)^{s(a+b)}) = SK_{AB}
\end{aligned}$$

instead. Evidently, the attack outlined in Figure 3 will no longer work since a non-matching conversation (i.e., $\mathcal{T}_A \neq \mathcal{T}_B$) will also mean that the session key is different, as shown below:

$$SK_{AB} = \mathcal{H}(K_{AB}) = \mathcal{H}(A||B||aQ_A||(b+c)Q_B||\hat{e}(S_A, W_B + aQ_B)),$$
$$SK_{BA} = \mathcal{H}(K_{BA}) = \mathcal{H}(A||B||(a+c)Q_A||bQ_B||\hat{e}(W_A + bQ_A, S_B)) \neq SK_{AB}.$$

Similarly, a reflection attack or an unknown key share attack would not work against the protocol since the construction of the session key introduces role asymmetry and the identities of the participants. In other words, session keys will be different when the roles of the same principal switch. Therefore, $\mathcal{A}$ appears to be unable to gain information about such fresh session key(s).

3.3 Sketch of New Proof

At first glance, it would seem that by fixing the attack outlined in Section 3.1, we have addressed the reasons why no Reveal query was allowed that was outlined in the existing proofs, and would be able to prove the improved protocol secure in the unrestricted BR93 model. However, we demonstrate that this is not possible unless we restrict the adversary from asking any Reveal queries to the partner of the Test session, as explained in Figure 4. However, by allowing the adversary to ask Reveal queries directed at the owner of the Test session (in our proof), we effectively prove the improved protocol secure against reflection attacks.

Recall that the general notion of the proof is to assume that there exists an adversary $\mathcal{A}$ who can gain a non-negligible advantage in distinguishing the test key in the game described in Section 2.4, and use $\mathcal{A}$ to break the underlying BDH problem. In other words, we build an adversary, $\mathcal{A}_{\mathcal{BDH}}$, against the BDH problem using $\mathcal{A}$. The objective of $\mathcal{A}_{\mathcal{BDH}}$ is to compute and output the value $\widehat{e}(P,P)^{xyz} \in G_2$ when given a bilinear map $\widehat{e}$, a generator of P of G_1, and a triple of elements $xP, yP, zP \in G_1$ with $x, y, z \in \mathbb{Z}_q^*$, where q is the prime order of the distinct groups G_1 and G_2.

Let oracle $\Pi^u_{A,B}$ be the initiator associated with the target Test session, and oracle $\Pi^v_{B,A}$ be the responder partner to $\Pi^u_{A,B}$. $\mathcal{A}_{\mathcal{BDH}}$ needs to simulate all responses to queries from $\mathcal{A}$, including the random oracle, $\mathcal{H}$. The proof specifies that $\mathcal{A}_{\mathcal{BDH}}$ can create all public/private key pairs for all players, except a randomly chosen player J. Let (Q_U, S_U) denote the public/private keys of players U other than J (where $S_U = xQ_U$). $\mathcal{A}_{\mathcal{BDH}}$ is unable to compute the private key of J because $\mathcal{A}_{\mathcal{BDH}}$ is trying to solve the BDH problem, which is embedded in the public key of J.

Figure 4 shows a possible sequence of adversary actions and the responses generated by $\mathcal{A}_{\mathcal{BDH}}$. It can be seen that $\mathcal{A}$ will be able to distinguish between the simulation provided by $\mathcal{A}_{\mathcal{BDH}}$ and the actual protocol if it carries out this sequence of actions, since with overwhelming probability, $v \neq SK_{BC}$ (recall that v is randomly chosen). Hence, $\mathcal{A}_{\mathcal{BDH}}$ cannot answer any Reveal directed at the partner of the target Test session.

$\mathcal{A}_{\mathcal{BDH}}$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\mathcal{A}$

$b \in_R \mathbb{Z}_r^*$ $\quad \xleftarrow{\mathsf{Send}(B, C, j, cQ_C)}$ $\qquad\qquad\qquad\qquad$ $c \in_R \mathbb{Z}_r^*$

$\xrightarrow{bQ_B}$

$\xleftarrow{\mathsf{Reveal}(B, C, j)}$

$\mathcal{A}_{\mathcal{BDH}}$ is supposed to respond with $\mathcal{H}(B||C||j||\hat{e}(cQ_C + bQ_C, S_B))$, but $\mathcal{A}_{\mathcal{BDH}}$ does not know S_B, and thus cannot know the input for its simulation of $\mathcal{H}$.

$v \in_R \{0,1\}^k$ $\quad \xrightarrow{v}$

$\xleftarrow{\mathsf{Corrupt}(C)}$

$\mathcal{A}_{\mathcal{BDH}}$ returns all internal states of C, including $S_C = sQ_C$.

$\xrightarrow{S_C}$ $\qquad$ Compute $SK_{BC} = \mathcal{H}(C||B||i||\hat{e}(S_C, bQ_B + cQ_B))$

Verify whether $v \overset{?}{=} SK_{BC}$

Fig. 4. An example simulation of Chen–Kudla protocol 2

Theorem 1. *The improved Chen–Kudla protocol 2 is a secure authenticated key establishment protocol in the sense of Definition 3 if the Bilinear Diffie-Hellman (BDH) problem is hard and the hash function, $\mathcal{H}$, is a random oracle, and the adversary $\mathcal{A}$ does not ask any* Reveal *queries to any sessions owned by the partner player associated with the* Test *session.*

The proof of Theorem 1 generally follows that of Chen & Kudla [10, Proof of Theorem 1], except that we allow $\mathcal{A}$ to ask Reveal queries (but not to the partner

Queries	Actions
$\mathsf{Send}(U_1, U_2, i)$	$\mathcal{A}_{\mathcal{BDH}}$ answers all Send queries in the same fashion as the proof simulation presented by Chen & Kudla.
$\mathsf{Corrupt}(U, K)$	$\mathcal{A}_{\mathcal{BDH}}$ answers all Corrupt queries in the same fashion as the proof simulation presented by Chen & Kudla.
$\mathsf{Test}(U_1, U_2, i)$	$\mathcal{A}_{\mathcal{BDH}}$ answers the Test query in the same fashion as the proof simulation presented by Chen & Kudla.
$\mathcal{H}(U_1\|\|U_2\|\|i\|\|te(m))$	$\mathcal{A}_{\mathcal{BDH}}$ will return a random value, $v \in_R \{0,1\}^k$ where k is the security parameter and store m in a list of tuples.
$\mathsf{Reveal}(U_1, U_2, i)$	If oracle $\Pi^i_{U_1,U_2}$ is not an oracle associated with the test session (or partner of such an oracle), and U_1 is not player J where $\mathcal{A}_{\mathcal{BDH}}$ did not generate the contents of the Send query to $\Pi^i_{U_1,U_2}$, then $\mathcal{A}_{\mathcal{BDH}}$ returns the associated session key. Otherwise $\mathcal{A}_{\mathcal{BDH}}$ terminates and halts the simulation. We observe that if $\mathcal{A}_{\mathcal{BDH}}$ halts because $U_1 = J$, the Test session chosen by $\mathcal{A}$ must be different to that desired by $\mathcal{A}_{\mathcal{BDH}}$, so even if the simulation had not halted here, it would have halted later.

Fig. 5. $\mathcal{A}_{\mathcal{BDH}}$ simulates the view of $\mathcal{A}$ by answering all Send, Reveal, Corrupt, and Test oracle queries of $\mathcal{A}$

player of the Test session). The details of the game simulation remain unchanged to that presented by Chen & Kudla [10, Proof of Theorem 1], except that we allow $\mathcal{A}$ to ask Reveal queries (but not to the partner player of the Test session), as given in Figure 5.

Hence, $\mathcal{A}_{\mathcal{BDH}}$ is able to simulate the view of $\mathcal{A}$ perfectly by answering all oracle queries of $\mathcal{A}$ as specified in Figure 5. Upon the conclusion of the game (i.e., $\mathcal{A}$ is done), $\mathcal{A}_{\mathcal{BDH}}$ chooses a random element in the list of tuples and outputs it. The probability that $\mathcal{A}_{\mathcal{BDH}}$ did not abort at some stage and produces the correct output remains non-negligible. This concludes the sketch of the proof of the theorem.

4 2P-IDAKA Protocol

In recent work, McCullagh & Barreto [18] proposed a two-party ID-based authenticated key agreement (2P-IDAKA) protocol with a proof of security in a weaker variant of the BR93 model whereby the adversary is not allowed to ask Reveal queries. Figure 6 describes the 2P-IDAKA protocol. There are two entities in the protocol, namely an initiator player A and a responder player B. Notation used in the protocols is as follows: $(s+a)P$ denotes the public key of A, $A_{pri} = ((s+a))^{-1}P$ denotes the private key of A, $(s+b)P$ denotes the public key of B, and $B_{pri} = ((s+b))^{-1}P$ denotes the private key of B. At the end of the protocol execution, both A and B accept session keys $SK_{AB} = \hat{e}(B_{KA}, A_{pri})^{x_a} = \hat{e}(P,P)^{x_a x_b} = SK_{BA}$.

$$\begin{array}{ccc}
A & & B \\
x_a \in_R Z_r^* & \xrightarrow{A_{KA} = x_a(s+b)P} & x_b \in_R Z_r^* \\
\hat{e}(B_{KA}, A_{pri})^{x_a} = \hat{e}(P,P)^{x_a x_b} & \xleftarrow{B_{KA} = x_b(s+a)P} & \hat{e}(A_{KA}, B_{pri})^{x_b} = \hat{e}(P,P)^{x_a x_b}
\end{array}$$

Fig. 6. McCullagh–Barreto 2P-IDAKA protocol

4.1 Why Reveal Query is Restricted

No Reveal query is allowed on the 2P-IDAKA protocol [11] due to the description provided in Figure 7.

In the protocol execution shown in Figure 7, both A and B have accepted the same session key (i.e., $SK_A = SK_B$). However, both A and B are non-partners since they do not have matching conversations as A's transcript is $(A_{KA}, B_{KA} \cdot x_E)$ whilst B's transcript is $(A_{KA} \cdot x_E, B_{KA})$. By sending a Reveal query to either A or B, $\mathcal{A}$ is able to trivially expose a fresh session key by asking a Reveal query to either A or B. Hence, the 2P-IDAKA protocol shown in Figure 6 is not secure since $\mathcal{A}$ is able to obtain the session key of a fresh oracle of a non-partner oracle by revealing a non-partner oracle holding the same key, in violation of the key establishment goal.

$$\begin{array}{ccccc}
A & & \mathcal{A} & & B \\
x_a \in_R \mathbb{Z}_r^* & \xrightarrow{A_{KA} = x_a(s+b)P} & \text{Intercept} & & \\
 & & x_E \in_E Z_r^* & & \\
 & & \text{Impersonate } A & \xrightarrow{A_{KA} \cdot x_E} & x_b \in_R \mathbb{Z}_r^* \\
 & & \text{Intercept} & \xleftarrow{B_{KA} = x_b(s+a)P} & \\
 & \xleftarrow{B_{KA} \cdot x_E} & \text{Impersonate } B & & \\
\multicolumn{5}{c}{SK_A = \hat{e}(x_b(s+a)P \cdot x_E, A_{pri})^{x_a} = \hat{e}(P,P)^{x_a x_b x_E} = SK_B}
\end{array}$$

Fig. 7. Execution of 2P-IDAKA protocol in the presence of a malicious adversary

4.2 Errors in Existing Proof

The general notion of the existing proof of McCullagh & Barreto [18, Proof of Theorem 1], to assume that there exists an adversary $\mathcal{A}$ who can gain a non-negligible advantage in distinguishing the test key in the game described in Section 2.4, and use $\mathcal{A}$ to break the underlying Bilinear Inverse Diffie–Hellman Problem (BIDHP). In other words, an adversary, $\mathcal{A}_{\mathcal{BIDHP}}$, against the BIDHP is constructed using $\mathcal{A}$. The objective of $\mathcal{A}_{\mathcal{BIDHP}}$ is to compute and output the value $\widehat{e}(P,P)^{\alpha^{-1}\beta}$ when given $P, \alpha P, \beta P$ for $x, y, z \in \mathbb{Z}_r^*$.

Error 1: In the existing proof, the public and private key pairs for some player, U_i, are selected as $((u-s)P$,$u^{-1}P)$, in contradiction to their description in the protocols where $((s+u)P$,$(s+u)^{-1}P)$ is given instead. The adversary, $\mathcal{A}$, is then able to tell that the public and private key pairs do not match by simply corrupting any player, as shown in Figure 8.

$$\begin{array}{ccc}
\mathcal{A}_{\mathcal{BIDHP}} & & \mathcal{A} \\
\text{Return all internal state of } U, & \xleftarrow{\mathsf{Corrupt}(U)} & \\
\text{including } (u)^{-1}P & \xrightarrow{u^{-1}P} & \text{Compute } \widehat{e}(uP - sP, (u)^{-1}P)
\end{array}$$

Fig. 8. Illustration of error 1

We can check whether a public and private key pair match by computing $\widehat{e}((s+u)P, (s+u)^{-1}P) = \widehat{e}((P,P)^{(s+u)(s+u)^{-1}} = \widehat{e}(P,P)$. However, as outlined in Figure 8, when $\mathcal{A}$ computes the public and private key pair of U, $\widehat{e}(uP - sP, (u)^{-1}P) = \widehat{e}((u-s)P, u^{-1}P) = \widehat{e}(P,P)^{(u-s)u^{-1}} = \widehat{e}(P,P)^{1-su^{-1}} \neq \widehat{e}(P,P)$. $\mathcal{A}$ trivially knows that the public and private key pairs of U do not match. Hence, the existing proof is invalidated.

Error 2: We observed that the parameter $\beta P = x_j \alpha P$ given in the existing proof should be $\beta P = x_j(y_i - s)P$ instead, as explained in Figure 9. In Figure 9, we assume that error 1 has been fixed. The public/private key pair of I (the partner player associated with the Test session is $((y_i - s)P, (y_i - s)^{-1}P)$, the public key of J (the owner of the Test session) is αP, and the private key of J (i.e., $\alpha^{-1}P$)) is unknown to both $\mathcal{A}_{\mathcal{BIDHP}}$ and $\mathcal{A}$. It is obvious from Figure 9 that we cannot

$$\begin{array}{lcl}
\mathcal{A}_{\mathcal{BIDHP}} & & \mathcal{A} \\
x_i \in_R \mathbb{Z}_r^* & \xleftarrow{\mathsf{Send}(J, I, i, 'start')} & \\
 & \xrightarrow{x_i P} & \mathcal{A} \text{ is given } \beta P = x_j \alpha P \text{ as input from } j \\
x_i P = x_t(\alpha P) & \xleftarrow{\mathsf{Send}(I, J, j, \beta P)} &
\end{array}$$

Fig. 9. Illustration of error 2

have the values of both $x_i P$ and $x_j P$ computed using the public key of J, αP (at least one of $x_i P$ and $x_j P$ have to be computed using the public key of I). To check, we compute $\widehat{e}(P,P)^{x_t x_j} = \widehat{e}(P,P)^{x_i \alpha^{-1} \beta \alpha^{-1}} \neq \widehat{e}(P,P)^{\alpha^{-1}\beta}$, which is what $\mathcal{A}_{\mathcal{BIDHP}}$ is trying to solve. Hence, the correct value for $\beta P = x_j \alpha P$ given in the existing proof should be $\beta P = x_j(y_i - s)P$ instead.

Further remarks: We observe that for the existing proof to work, we would have to assume that the inputs to the Test session originated with the simulator, $\mathcal{A}_{\mathcal{BIDHP}}$, and not the adversary, $\mathcal{A}$. However, this is not a normal assumption and resricts the BR93 model. In fact, if a slightly different assumption were made in the proof of the improved Chen & Kudla's protocol in Section 3.3, namely that if B is the partner of the Test session, then all Send query inputs to sessions of B that are later revealed were generated by $\mathcal{A}_{\mathcal{BDH}}$, then the proof in Section 3.3 would not have to restrict Reveal queries to B.

Consequences of errors in security proofs: Protocol implementers (usually non-specialists and/or industrial practitioners) will usually plug-and-use existing provably-secure protocols without reading the formal proofs of the protocols [16]. Errors in security proofs or specifications themselves certainly will certainly undermine the credibility and trustworthiness of provably-secure protocols in the real world.

4.3 Improved 2P-IDAKA Protocol

Let A's transcript be denoted by $\mathcal{T}_A$ and B's transcript be denoted by $\mathcal{T}_B$. Consider the scenario whereby session keys of A and B are constructed as

$$SK_{AB} = \mathcal{H}(A||B||\mathcal{T}_A||\hat{e}(B_{KA}, A_{pri})^{x_a}) = \mathcal{H}(A||B||\mathcal{T}_A||\hat{e}(P,P)^{x_a x_b}),$$
$$SK_{BA} = \mathcal{H}(A||B||\mathcal{T}_B||\hat{e}(A_{KA}, B_{pri})^{x_b}) = \mathcal{H}(A||B||\mathcal{T}_B||\hat{e}(P,P)^{x_a x_b}) = SK_{AB}$$

instead. Evidently, the attack outlined in Figure 7 will no longer be valid since a non-matching conversation (i.e., $\mathcal{T}_A \neq \mathcal{T}_B$) will also mean that the session key is different, as shown below:

$$SK_{AB} = \mathcal{H}(A||B||x_a(s+b)P||(x_b \cdot x_E)(s+a)P||\hat{e}(B_{KA}, A_{pri})^{x_a}),$$
$$SK_{BA} = \mathcal{H}(A||B||(x_a \cdot x_E)(s+b)P||x_b(s+a)P||\hat{e}(A_{KA}, B_{pri})^{x_b}) \neq SK_{AB}.$$

Therefore, $\mathcal{A}$ is unable to gain information about any fresh session key(s). Figure 10 illustrates why Reveal queries directed at the owner of the Test session cannot be answered by $\mathcal{A}_{\mathcal{BDH}}$. Note that $\Pi^j_{J,C}$ is not the target Test session.

$\mathcal{A}_{\mathcal{BIDHP}}$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\mathcal{A}$

$x_b \in_R \mathbb{Z}_r*$ $\quad \xleftarrow{\mathsf{Send}(J, C, j, (x_c(s+b)P))}$ $\quad x_c \in_R \mathbb{Z}_r*$

$\xrightarrow{(x_b(s+b)P)}$

$\xleftarrow{\mathsf{Reveal}(J, C, j)}$

$\mathcal{A}_{\mathcal{BIDHP}}$ is supposed to respond with $\mathcal{H}(J||C||j||\hat{e}(x_c(s+b)P, J_{pri}))$, but $\mathcal{A}_{\mathcal{BIDHP}}$ does not know J_{pri}, and thus cannot know the input for its simulation of $\mathcal{H}$.

$v \in_R \{0,1\}^k$ $\quad \xrightarrow{v}$

$\xleftarrow{\mathsf{Corrupt}(C)}$

$\mathcal{A}_{\mathcal{BIDHP}}$ returns all internal states of C, including $C_{pri} = (s+c)^{-1}P$.

$\xrightarrow{C_{pri}}$ $\quad SK_{BC} = \mathcal{H}(C||B||i||\hat{e}(x_c(s+b)P, C_{pri}))$

Verify if $v \overset{?}{=} SK_{BC}$

Fig. 10. An example simulation of McCullagh–Barreto 2P-IDAKA protocol

From Figure 10, it can be seen that $\mathcal{A}$ will be able to distinguish between the simulation provided by $\mathcal{A}_{\mathcal{BIDHP}}$ and the actual protocol if it carries out this sequence of actions, since with overwhelming probability, $v \neq SK_{BC}$ (recall that v is randomly chosen). Hence, $\mathcal{A}_{\mathcal{BIDHP}}$ cannot answer any Reveal directed at the owner of the target Test session, J, unless we made a similar type of assumption in the existing proof outlined in Section 4.2 that all Send query inputs to sessions of J that are later revealed were generated by $\mathcal{A}_{\mathcal{BIDHP}}$.

5 A Proposal for Session Key Construction

In this section, we present our proposal on how session keys should be constructed. Although we do not claim that session keys constructed in this fashion will result in a secure protocol (as the security of the protocol is based on many other factors, such as the underlying cryptographic primitives used), we do claim that having a sound construction of session keys may reduce the number of possible attacks on the protocol.

We propose that session keys in key establishment protocols should be constructed in the following fashion, as shown in Table 2. The inclusion of

– the identities of the participants and their roles provides resilience against unknown key share attacks and reflection attacks since the inclusion of both the identities of the participants and role asymmetry effectively ensures some sense of direction. If the role of the participants or the identities of the (perceived) partner participants change, the session keys will also be different,
– the unique session identifiers (SIDs) ensures that session keys will be fresh, and if SIDs are defined as the concatenation of messages exchanged during the protocol execution, messages altered during the transmission will result in different session keys (providing data origin authentication), and
– some other ephemeral shared secrets and/or long-term (static) shared secrets depending on individual protocols, ensures that the session key is only known to the protocol participants.

Table 2. Construction of session key in key establishment protocols

Session key input	Properties
Identities of the participants and their roles	Resilience against unknown key share attacks [8, Chapter 5.1.2] and reflection attacks [17].
Unique session identifiers (SIDs)	Freshness and data origin authentication (assuming SIDs defined to be the concatenation of exchanged messages).
Ephemeral shared secrets and/or long-term (static) shared secrets	If the identities of the (perceived) partner participants change, the session keys will also be different.

6 Conclusion

By making a small change to the way session keys are constructed in the Chen–Kudla protocol 2 and McCullagh–Barreto protocol 2P-IDAKA, we demonstrated that the existing attacks no longer work. In addition, both protocols' proof were improved to be less restrictive with regard to the Reveal queries allowed[1]. We also found some errors in the McCullagh–Barreto proof, as well as observing that it is in a restricted version of the BR93 model that assumes that the adversary does not generate the input to the Test session.

As a result of our findings, we would recommend that all provably secure protocols should construct session keys using materials comprising the identities of the participants and roles, unique session identifiers (SIDs), and some other

[1] Chow [13] pointed out that the technicality of not being able to answer Reveal queries outlined in Sections 3.3 and 4.3 can be resolved using GAP assumptions [19].

ephemeral shared secrets and/or long-term (static) shared secrets. We hope that this work contributes towards a better understanding on how to construct secure session keys in key establishment protocols.

References

1. Feng Bao. Security Analysis of a Password Authenticated Key Exchange Protocol. In Colin Boyd and Wenbo Mao, editors, *6th Information Security Conference - ISC 2003*, pages 208–217. Springer-Verlag, 2003. Volume 2851/2003 of Lecture Notes in Computer Science.
2. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A Modular Approach to The Design and Analysis of Authentication and Key Exchange Protocols. In Jeffrey Vitter, editor, *30th ACM Symposium on the Theory of Computing - STOC 1998*, pages 419–428. ACM Press, 1998.
3. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, pages 139 – 155. Springer-Verlag, 2000. Volume 1807/2000 of Lecture Notes in Computer Science.
4. Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology - Crypto 1993*, pages 110–125. Springer-Verlag, 1993. Volume 773/1993 of Lecture Notes in Computer Science.
5. Mihir Bellare and Phillip Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In F. Tom Leighton and Allan Borodin, editors, *27th ACM Symposium on the Theory of Computing - STOC 1995*, pages 57–66. ACM Press, 1995.
6. Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key Agreement Protocols and their Security Analysis. In Michael Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, pages 30–45. Springer-Verlag, 1997. Volume 1355/1997 of Lecture Notes in Computer Science.
7. Simon Blake-Wilson and Alfred Menezes. Security Proofs for Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques. In Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe, editors, *Security Protocols Workshop*, pages 137–158. Springer-Verlag, 1997. Volume 1361/1997 of Lecture Notes in Computer Science.
8. Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment.* Springer-Verlag, June 2003.
9. Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels (Extended version available from `http://eprint.iacr.org/2001/040/`). In Birgit Pfitzmann, editor, *Advances in Cryptology - Eurocrypt 2001*, pages 453–474. Springer-Verlag, 2001. Volume 2045/2001 of Lecture Notes in Computer Science.
10. Liqun Chen and Caroline Kudla. Identity Based Authenticated Key Agreement Protocols from Pairings (Corrected version at `http://eprint.iacr.org/2002/184/`). In *16th IEEE Computer Security Foundations Workshop - CSFW 2003*, pages 219–233. IEEE Computer Society Press, 2003.
11. Kim-Kwang Raymond Choo. Revisit Of McCullagh–Barreto Two-Party ID-Based Authenticated Key Agreement Protocols. Cryptology ePrint Archive, Report 2004/343, 2004. `http://eprint.iacr.org/2004/343/`.

12. Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. The Importance of Proofs of Security for Key Establishment Protocols: Formal Analysis of Jan–Chen, Yang–Shen–Shieh, Kim–Huh–Hwang–Lee, Lin–Sun–Hwang, & Yeh–Sun Protocols (Extended version available from `http://eprints.qut.edu.au/perl/user_eprints?userid=51`). *(To appear in) Journal of Computer Communications - Special Issue of Internet Communications Security*, 2005.
13. Sherman S. M. Chow. Personal Communication, 29 Apr 2005.
14. Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in Key Distribution Protocols. *ACM Journal of Communications*, 24(8):533–536, 1981.
15. Ik Rae Jeong, Jonathan Katz, and Dong Hoon Lee. One-Round Protocols for Two-Party Authenticated Key Exchange. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *Applied Cryptography and Network Security - ACNS 2004*, pages 220–232. Springer-Verlag, 2004. Volume 3089/2004 of Lecture Notes in Computer Science.
16. Neal Koblitz and Alfred Menezes. Another Look at "Provable Security". Technical report CORR 2004-20, Centre for Applied Cryptographic Research, University of Waterloo, Canada, 2004.
17. Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. In Dan Boneh, editor, *Advances in Cryptology - Crypto 2003*, pages 400–425. Springer-Verlag, 2003. Volume 2729/2003 of Lecture Notes in Computer Science.
18. Noel McCullagh and Paulo S. L. M. Barreto. A New Two-Party Identity-Based Authenticated Key Agreement (Extended version available from `http://eprint.iacr.org/2004/122/`). In Alfred John Menezes, editor, *Cryptographers' Track at RSA Conference - CT-RSA 2005*, pages 262–274. Springer-Verlag, 2005. Volume 3376/2005 of Lecture Notes in Computer Science.
19. Tatsuaki Okamoto and David Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In Kwangjo Kim, editor, *2001 International Workshop on Practice and Theory in Public Key Cryptography - PKC 2001*. Springer-Verlag, 2001. Volume 1992/2001 of Lecture Notes in Computer Science.
20. Olivier Pereira and Jean-Jacques Quisquater. Some Attacks Upon Authenticated Group Key Agreement Protocols. *Journal of Computer Security*, 11:555–580, 2003.

On the Security of Probabilistic Multisignature Schemes and Their Optimality

Yuichi Komano[1], Kazuo Ohta[2], Atsushi Shimbo[1], and Shinichi Kawamura[1]

[1] Toshiba Corporation,
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan
{yuichi1.komano,atsushi.shimbo,shinichi2.kawamura}@toshiba.co.jp
[2] The University of Electro-Communications,
Chofugaoka 1-5-1, Chofu-shi, Tokyo 182-8585, Japan
ota@ice.uec.ac.jp

Abstract. We first prove that the following three probabilistic multisignature schemes based on a trapdoor permutation have tight security; PFDH (probabilistic full domain hash) based multisignature scheme (PFDH-MSS), PSS (probabilistic signature scheme) based multisignature scheme (PSS-MSS), and short signature PSS based multisignature scheme (S-PSS-MSS). Second, we give an optimal proof (general result) for multisignature schemes, which derives the lower bound for the length of random salt. We also estimate the upper bound for the length in each scheme and derive the optimal length of a random salt. Two of the schemes are promising in terms of security tightness and optimal signature length.

Keywords: Multisignature, Aggregate signature, Provable security, Optimal security, Random oracle model

1 Introduction

1.1 Background

The notion of multisignatures was derived by Itakura and Nakamura [5], and a great deal of research has been done on this subject. In multisignature schemes, two or more signers generate one multisignature for some message: the same result is accomplished by concatenating each signer's signature; however, the multisignature scheme decreases the (total) length of the signature and/or the signing (verification) costs.

In respect of provably secure multisignatures (in the sense of the security model of [9]) based on a trapdoor (one-way) permutation (*e.g.*, RSA), Mitomi and Miyaji in Appendix A of [10] and Kawauchi and Tada [6] proposed FDH based multisignature scheme and probabilistic multisignature scheme based on PSS, respectively. In these schemes, the signing order is restricted by a key length of each signer. Moreover, as the signing order proceeds, the computation cost is enlarged (because of the increase of key length).

With regard to optimal security and optimal length of a random salt utilized in probabilistic signature schemes, Coron [4] introduced the notion of "reduction" and claimed that (1) the security of PSS is not improved if the length of a random salt exceeds a certain value (upper bound), (2) a lower bound for the length of a random salt, with which we guarantee the security of probabilistic signature schemes (general result), is derived from the optimal proof utilizing the notion of the "reduction", and, (3) the upper bound (of PSS) derived from (1) equals the lower bound derived from (2); which gives the optimal length of a random salt.

1.2 Our Contribution

This paper deals with three multisignature schemes based on the trapdoor permutation; PFDH (probabilistic full domain hash, [4]) based multisignature scheme (PFDH-MSS, [7]), PSS (probabilistic signature scheme, [2]) based multisignature scheme (PSS-MSS), and short signature PSS based multisignature scheme (S-PSS-MSS). The construction of PFDH-MSS and PSS-MSS is a similar to that of the sequential signature scheme (hereafter, we call the scheme FDH-MSS[1]) based on the full domain hash (FDH, [1]) constructed by Lysyanskaya et al. [8]. For simplicity, we give a description of the schemes with the RSA function [11] as the trapdoor permutation in section 3.

We first prove that these three multisignature schemes have tight security under the random oracle model [1]. We then show that PFDH-MSS and S-PSS-MSS have optimal length of signature; an increase of signature size per signer is the same as the length of a random salt. Since the random salt is necessary for ensuring tight security and the salt should be recovered in the verification step, it is inevitable that the length of signature is enlarged by more than or equal to the length of random salt per signer; namely, the signature size in each scheme is optimal.

Second, we apply Coron's technique (using "reduction") to the multisignature schemes and estimate the optimal length of a random salt utilized in the schemes. We first show an optimal proof (general result) which derives the lower bound for the length of a random salt with which we ensure the security of multisignature scheme. Then, we prove the security of the schemes and derive the optimal length of a random salt. It is of theoretical interest that the optimal length of a random salt utilized in the multisignature schemes is equal to the optimal length of a random salt utilized in PSS (estimated in [4]).

1.3 Related Work – Sequential Aggregate Signatures

In 2003, Boneh et al. [3] proposed an aggregate signature scheme which is recognized as one of the generalizations of the multisignature scheme. The aggregate signature scheme is a signature scheme which can unify several signatures generated by plural signers on different messages. The original aggregate signature

[1] Sequential aggregate signatures are essentially the same as message flexible multisignature schemes. See the section 1.3 for detail.

scheme [3] requires the restricted assumptions (GDH groups and/or bilinear maps, *e.g.*, Weil and Tate pairings on elliptic curves).

Recently, Lysyanskaya et al. [8] proposed a sequential aggregate signature scheme based on a more general function, trapdoor permutation (*e.g.*, RSA); however, since their scheme is based on FDH, the security is not tight enough.

The difference between the definition of an original multisignature scheme and that of a sequential aggregate signature scheme is: in multisignature schemes, signers sign the same message, and in sequential aggregate signature schemes, signers sign different messages. For multisignature schemes, in order to allow a signer to sign a different message, *message flexible* multisignature schemes have been invented. For aggregate signature schemes, Boneh et al. [3] claim that the signer can sign the same message by concatenating the message with some additional data (*e.g.*, identification number). Namely, the sequential aggregate signature is essentially the same as the message flexible multisignature scheme.

With regard to security model, the model described in [8], *the sequential aggregate chosen key model*, is more restricted than the security model of [9], in terms of not allowing a forger to select a victim. In this paper, we follow the security model of [9] to ensure the security of multisignature schemes.

2 Definitions

This section formalizes the definitions of a message flexible multisignature scheme (a sequential aggregate signature scheme) and its security model following those of [9]. Hereafter, we assume that the total group of signers $\mathcal{G}$ consists of (constant ordered) L players (signers), $P_1, P_2, \cdots$, and P_L who have identification numbers, $ID_1, ID_2, \cdots$, and ID_L, respectively, and also assume that L' signers $P_{i_1}, P_{i_2}, \cdots$, and $P_{i_{L'}} \in \mathcal{G}' \subseteq \mathcal{G}$ execute a multisigning algorithm. Note that these L' signers may be selected adaptively in the process of the multisigning algorithm.

Definition 1 (Multisignature Scheme). *A multisignature scheme consists of the following three algorithms, $(\mathcal{K}, \mathcal{MS}, \mathcal{V})$.*

— *Key generation algorithm $\mathcal{K}$ is a probabilistic algorithm which, given a security parameter k for each signer P_{i_j} of $\mathcal{G}$, outputs a key pair (public and private keys), $\mathcal{K}(1^k) = (\mathsf{pk}_{i_j}, \mathsf{sk}_{i_j})$.*

— *Multisigning algorithm $\mathcal{MS}$ is performed by $P_{i_j} \in \mathcal{G}'$. The inputs of this algorithm are message m_{i_j} (concatenated with the previous signer's message $m_{i_{j-1}}$), a signature of $P_{i_{j-1}}$ $\sigma_{i_{j-1}}$ and secret key sk_{i_j}, and the output is a signature $\sigma_{i_j} = \mathcal{MS}_{\mathsf{sk}_{i_j}}(m_{i_j}, \sigma_{i_{j-1}})$. This algorithm may be probabilistic.*

— *Verification algorithm $\mathcal{V}$ takes message $m_{i_{L'}}$, multisignature $\sigma_{i_{L'}}$, and public keys of signers $\mathsf{pk}_{i_1}, \cdots, \mathsf{pk}_{i_{L'}}$, and returns $\mathcal{V}_{\mathsf{pk}_{i_1}, \cdots, \mathsf{pk}_{i_{L'}}}(m_{i_{L'}}, \sigma_{i_{L'}}) = 1$ if $\sigma_{i_{L'}}$ is a valid signature of $m_{i_{L'}}$ and $\mathcal{G}'$, and otherwise, returns 0 (*Reject*). This algorithm is deterministic.*

Definition 2 (Security Model [9]). *We assume an attack model of a forger $\mathcal{F}$ of multisignature schemes as follows:*

1. *Key generation phase attack: In a key generation phase, $\mathcal{F}$ can corrupt any signer of $\mathcal{G}$ by his choice, i.e., $\mathcal{F}$ can request any signer to reveal his secret key (we call this request a corrupt query). Moreover, $\mathcal{F}$ can join the group by masquerading as a signer(s), i.e., $\mathcal{F}$ generates a pair (pairs) of public and secret keys and registers the public key(s), after $\mathcal{F}$ receives some key pairs (we call this request a masquerading query).*
2. *$\mathcal{F}$ receives the public keys of all signers (including the signers corrupted by $\mathcal{F}$ or whom $\mathcal{F}$ masquerades as).*
3. *Signing phase attack: In a signing phase, $\mathcal{F}$ can also corrupt any signer of $\mathcal{G}$ by his choice. Moreover, $\mathcal{F}$ can request any signer of his choice, except[2] those legal signers whom $\mathcal{F}$ masquerades as, to sign on adaptively chosen message m.*
4. *$\mathcal{F}$ outputs a forgery $(m^*_{i_{L'}}, \sigma^*_{i_{L'}})$.*

Let $\mathcal{C}$ be a group of signers who are corrupted by $\mathcal{F}$ or whom $\mathcal{F}$ masquerades as in the key generation phase attack or the signing phase attack. The forger's success probability is defined as:

$$\epsilon = \Pr[\mathcal{V}_{\mathsf{pk}_{i_1},\cdots,\mathsf{pk}_{i_{L'}}}(m^*_{i_{L'}}, \sigma^*_{i_L}) = 1 \wedge \exists j \in \{1, \cdots, L\} \text{ such that } P_{i_j} \notin \mathcal{C} \text{ and } P_{i_j} \text{ is not requested by } \mathcal{F} \text{ to sign } m^*_{i_j}].$$

We claim that a multisignature scheme is $(\tau, q_\Sigma, q_H, \epsilon)$-secure in accordance with Existentially Un-Forgeable against Adaptive Chosen Message Attack and Adaptive Insider Attack (EUF-ACMA&AIA) if arbitrary forger, whose running time is bounded by τ, cannot achieve a success probability more than ϵ after making at most $L-1$ corrupt and masquerading queries in total, q_{Σ_i} signing queries to i-th signing oracle Σ_i ($i = 1, 2, \cdots, L$, $q_\Sigma = q_{\Sigma_1} + \cdots + q_{\Sigma_L}$), and q_H hash queries to hash function H.

Also note that, in the multisignature scheme in which each signer generates a signing key independently of other signer (including the multisignature scheme based on RSA in which each signer utilizes her own modulus), signers can join the signing group at any time. In the model, we allow $\mathcal{F}$ to make a masquerading query at any time[3] (at the key generation and signing phases).

3 Probabilistic Multisignature Schemes

In this section, we first give the description of PFDH based multisignature scheme following [7], and then propose two PSS based multisignature schemes based on a *general* trapdoor permutation. For simplicity, we utilize the RSA function as the trapdoor permutation. Note that the security of these schemes can be ensured under the assumption of the general trapdoor permutation.

[2] $\mathcal{F}$ can request signing queries from signers including the signers corrupted by $\mathcal{F}$ in key phase and signing phase attacks.

[3] In some multisignature schemes (*e.g.*, [9]) in which the key generation phase is executed for each subgroup $\mathcal{G}'$, $\mathcal{F}$ can make a masquerading query only at the key generation phase.

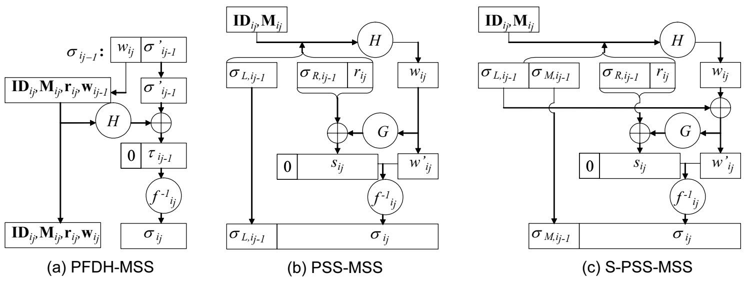


Fig. 1. Probabilistic Multisignature Schemes

3.1 PFDH Based Multisignature Scheme

We give the description of PFDH based multisignature scheme (PFDH-MSS). Its security is discussed in section 5. The only difference from the original (FDH-based) sequential aggregate signature [8] (FDH-MSS) is that a random salt is utilized in PFDH-MSS; which ensures tight security[4].

Protocol 1 (RSA-PFDH-MSS). *RSA-PFDH-MSS with hash function* $H : \{0,1\}^* \rightarrow \{0,1\}^{k-1}$ *is executed as follows (Figure 1 (a)):*

— *Key generation algorithm* $\mathcal{K}$*, given a security parameter* k*, outputs an RSA key pair*[5] *of each signer* P_{i_j} $(i_j = 1, \cdots, L)$*,* $\mathcal{K}(1^k) = ((n_{i_j}, e_{i_j}), d_{i_j})$.

— *Assume that signer* P_{i_j} *in* $\mathcal{G}' = \{P_{i_1}, \cdots P_{i_{L'}}\} \subseteq \mathcal{G}$ *signs on a message* m_{i_j}. P_{i_j}*, given identification number* $\mathbf{ID}_{i_{j-1}}$*, message* $\mathbf{M}_{i_{j-1}}$*, random salt* $\mathbf{r}_{i_{j-1}}$*, and signatures* $\mathbf{w}_{i_{j-1}}$ *and* $\sigma_{i_{j-1}}$ *from a previous signer* $P_{i_{j-1}}$ *(*$\mathbf{ID}_{i_0} = \mathbf{M}_{i_0} = \mathbf{r}_{i_0} = \mathbf{w}_{i_0} =$ *null and* $\sigma_{i_0} = 0^{k-1}$*), first chooses* $r_{i_j} \overset{R}{\leftarrow} \{0,1\}^{k_0}$ *and divides* $\sigma_{i_{j-1}}$ *(*k *bits) into two parts; lower* $k-1$ *bits,* $\sigma'_{i_{j-1}}$*, and the remaining part,* w_{i_j} *(MSB). Then,* P_{i_j} *sets* $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}}||ID_{i_j}$*,* $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}}||m_{i_j}$*,* $\mathbf{r}_{i_j} = \mathbf{r}_{i_{j-1}}||r_{i_j}$*, and* $\mathbf{w}_{i_j} = \mathbf{w}_{i_{j-1}}||w_{i_j}$*, and computes* $\tau_{i_j} = H(\mathbf{ID}_{i_j}, \mathbf{M}_{i_j}, \mathbf{r}_{i_j}, \mathbf{w}_{i_j}) \oplus \sigma'_{i_{j-1}}$ *and* $\sigma_{i_j} = (0||\tau_{i_j})^{d_{i_j}} (\mathrm{mod} n_{i_j})$*. Finally,* P_{i_j} *gives* $\mathbf{ID}_{i_j}$*,* $\mathbf{M}_{i_j}$*,* $\mathbf{r}_{i_j}$*,* $\mathbf{w}_{i_j}$*, and* σ_{i_j} *to the next signer.*

— *For* $\mathbf{ID}_{i_{L'}}$*,* $\mathbf{M}_{i_{L'}}$*,* $\mathbf{r}_{i_{L'}}$*,* $\mathbf{w}_{i_{L'}}$*, and* $\sigma_{i_{L'}}$*, we verify the validity of the signature by recovering* σ_{i_j} $(j = L', \cdots, 0)$ *and checking the validity of* σ_{i_0} *as follows: for* $\mathbf{ID}_{i_j}$*,* $\mathbf{M}_{i_j}$*,* $\mathbf{r}_{i_j}$*,* $\mathbf{w}_{i_j}$*, and* σ_{i_j}*, we first compute* $b||\tau_{i_j} = (\sigma_{i_j})^{e_{i_j}} (\mathrm{mod} n_{i_j})$*, where*[6] $|b| = 1$ *and* $|\tau_{i_j}| = k-1$*. If* $b \neq 0$*, we reject the signature. Otherwise, we recover* $\sigma'_{i_{j-1}} = H(\mathbf{ID}_{i_j}, \mathbf{M}_{i_j}, \mathbf{r}_{i_j}, \mathbf{w}_{i_j}) \oplus \tau_{i_j}$ *(the lower* $k-1$ *bit). Then, we divide* $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}}||ID_{i_j}$*,* $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}}||m_{i_j}$*,* $\mathbf{r}_{i_j} = \mathbf{r}_{i_{j-1}}||r_{i_j}$*, and* $\mathbf{w}_{i_j} =$

[4] In [7], Kawauchi and Tada constructed a PFDH based multisignature scheme and tried to give a security consideration of the scheme; however, their proof strategy has problems in simulating the answers to oracle queries. Moreover, they did not derive the optimal length of a random salt at all.

[5] n_{i_j} is a product of two (secret) large primes, and $\gcd(e_{i_j}, \phi(n_{i_j})) = 1$ and $e_{i_j} d_{i_j} \equiv 1 (\mathrm{mod} \phi(n_{i_j}))$ hold. Here, ϕ is Euler's phi function.

[6] Here, $|x|$ denotes the bit length of x.

$\mathbf{w}_{i_{j-1}}||w_{i_j}$, and recover $\sigma_{i_{j-1}} = w_{i_j}||\sigma'_{i_{j-1}}$, and execute the verification for i_{j-1}-th signer's signature.
Finally, if $\sigma_{i_0} = 0^{k-1}$, we accept the signature. Otherwise, we reject it.

Note that (RSA-)PFDH-MSS has tight security as we will see in section 5. The increase of length of signature per signer (including a random salt) is $k_0 + 1$ bits, where k_0 denotes the length of a random salt.

3.2 PSS Based Multisignature Schemes

In this subsection, we propose two PSS based multisignature schemes; the first one is naturally constructed from PSS (PSS-MSS), and the other one is constructed from PSS-MSS in order to decrease the signature size (S-PSS-MSS). We first give the description of RSA-PSS-MSS.

Protocol 2 (RSA-PSS-MSS). *RSA-PSS-MSS with hash functions $G : \{0,1\}^{k_1} \rightarrow \{0,1\}^{k-k_0-k_1-1}$, and $H : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$ is executed as follows (Figure 1 (b)):*

— *Key generation algorithm $\mathcal{K}$ is the same as that described in the Protocol 1.*
— *Assume that signer P_{i_j} in $\mathcal{G}' = \{P_{i_1}, \cdots P_{i_{L'}}\} \subseteq \mathcal{G}$ signs on a message m_{i_j}. P_{i_j}, given identification number $\mathbf{ID}_{i_{j-1}}$, message $\mathbf{M}_{i_{j-1}}$, and signature $\sigma_{i_{j-1}}$ from a previous signer $P_{i_{j-1}}$ ($\mathbf{ID}_{i_0} = \mathbf{M}_{i_0} =$ null and $\sigma_{i_0} = 0^{k-k_0-k_1-1}$), first chooses $r_{i_j} \overset{R}{\leftarrow} \{0,1\}^{k_0}$ and divides $\sigma_{i_{j-1}}$ ($k + (j-1)(k_0 + k_1 + 1)$ bits) into two parts; lower $k - k_0 - k_1 - 1$ bits, $\sigma_{R,i_{j-1}}$, and the remaining part, $\sigma_{L,i_{j-1}}$. Then, P_{i_j} sets $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}}||ID_{i_j}$ and $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}}||m_{i_j}$, and computes $w_{i_j} = H(\mathbf{ID}_{i_j}, \mathbf{M}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j})$ and $s_{i_j} = (\sigma_{R,i_{j-1}}||r_{i_j}) \oplus G(w_{i_j})$. P_{i_j} generates a signature $\sigma'_{i_j} = (0||s_{i_j}||w_{i_j})^{d_{i_j}} (\mathrm{mod} n_{i_j})$. Finally, P_{i_j} gives $\mathbf{ID}_{i_j}$, $\mathbf{M}_{i_j}$, and $\sigma_{i_j} = \sigma_{L,i_{j-1}}||\sigma'_{i_j}$ to the next signer.*
— *For $\mathbf{ID}_{i_{L'}}$, $\mathbf{M}_{i_{L'}}$, and $\sigma_{i_{L'}}$, we verify the validity of the signature by recovering σ_{i_j} ($j = L', \cdots, 0$) and checking the validity of σ_{i_0} as follows: for $\mathbf{ID}_{i_j}$, $\mathbf{M}_{i_j}$, and σ_{i_j}, we first divide σ_{i_j} into two parts; lower k bits, σ'_{i_j}, and the remaining part, $\sigma_{L,i_{j-1}}$. Then, we compute $b||s_{i_j}||w_{i_j} = (\sigma'_{i_j})^{e_{i_j}} (\mathrm{mod} n_{i_j})$, where $|b| = 1$, $|s_{i_j}| = k - k_1 - 1$, and $|w_{i_j}| = k_1$. If $b \neq 0$, we reject the signature. Otherwise, we recover $\sigma_{R,i_{j-1}}||r_{i_j} = s_{i_j} \oplus G(w_{i_j})$, where $|\sigma_{R,i_{j-1}}| = k - k_0 - k_1 - 1$ and $|r_{i_j}| = k_0$. Then, we compute $w'_{i_j} = H(\mathbf{ID}_{i_j}, \mathbf{M}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j})$ and reject the signature if $w_{i_j} \neq w'_{i_j}$. Otherwise, we divide $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}}||ID_{i_j}$ and $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}}||m_{i_j}$, and recover $\sigma_{i_{j-1}} = \sigma_{L,i_{j-1}}||\sigma_{R,i_{j-1}}$, and execute the verification for i_{j-1}-th signer's signature.*
Finally, if $\sigma_{i_0} = 0^{k-k_0-k_1-1}$, we accept the signature. Otherwise, we reject it.

From the Protocol 2, the increase of length of signature per signer in PSS-MSS is $k_0 + k_1 + 1$ bits, where k_0 and k_1 are the length of a random salt and the output length of hash function G, respectively. The increase in PSS-MSS is greater than that in PFDH-MSS by k_1 bits per signer.

Now, we construct short signature PSS based multisignature scheme (S-PSS-MSS) by improving PSS-MSS. Broadly speaking, the strategy of constructing

S-PSS-MSS is that we divide $\sigma_{L,i_{j-1}}$ in PSS-MSS into two parts, $\sigma_{L,i_{j-1}}$ and $\sigma_{M,i_{j-1}}$, and exclusive-or $\sigma_{L,i_{j-1}}$ with $H(\mathbf{ID}_{i_j}, \mathbf{M}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j})$. We give the formal description of RSA-S-PSS-MSS.

Protocol 3 (RSA-S-PSS-MSS). *RSA-S-PSS-MSS with hash functions* $G : \{0,1\}^{k_1} \rightarrow \{0,1\}^{k-k_0-k_1-1}$, *and* $H : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$ *is executed as follows (Figure 1 (c)):*

— *Key generation algorithm* $\mathcal{K}$ *is the same as that described in the Protocol 1.*
— *Assume that signer* P_{i_j} *in* $\mathcal{G}' = \{P_{i_1}, \cdots P_{i_{L'}}\} \subseteq \mathcal{G}$ *signs on a message* m_{i_j}. P_{i_j}, *given identification number* $\mathbf{ID}_{i_{j-1}}$, *message* $\mathbf{M}_{i_{j-1}}$, *and signature* $\sigma_{i_{j-1}}$ *from a previous signer* $P_{i_{j-1}}$ ($\mathbf{ID}_{i_0} = \mathbf{M}_{i_0} = \mathsf{null}$ *and* $\sigma_{i_0} = 0^{k-k_0+k_1-1}$), *first chooses* $r_{i_j} \overset{R}{\leftarrow} \{0,1\}^{k_0}$ *and divides* $\sigma_{i_{j-1}}$ ($k+(j-1)(k_0+1)$ *bits) into three parts; the least significant* $k-k_0-k_1-1$ *bits,* $\sigma_{R,i_{j-1}}$, *the most significant* k_1 *bits,* $\sigma_{L,i_{j-1}}$, *and the remaining part,* $\sigma_{M,i_{j-1}}$. *Then,* P_{i_j} *sets* $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}}||ID_{i_j}$ *and* $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}}||m_{i_j}$, *and computes* $w'_{i_j} = H(\mathbf{ID}_{i_j}, \mathbf{M}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j})$, $w_{i_j} = \sigma_{L,i_{j-1}} \oplus w'_{i_j}$, *and* $s_{i_j} = (\sigma_{R,i_{j-1}}||r_{i_j}) \oplus G(w_{i_j})$. P_{i_j} *generates a signature* $\sigma'_{i_j} = (0||s_{i_j}||w_{i_j})^{d_{i_j}} (\mathrm{mod} n_{i_j})$. *Finally,* P_{i_j} *gives* $\mathbf{ID}_{i_j}$, $\mathbf{M}_{i_j}$, *and* $\sigma_{i_j} = \sigma_{M,i_{j-1}}||\sigma'_{i_j}$ *to the next signer.*
— *For* $\mathbf{ID}_{i_{L'}}$, $\mathbf{M}_{i_{L'}}$, *and* $\sigma_{i_{L'}}$, *we verify the validity of the signature by recovering* σ_{i_j} ($j = L', \cdots, 0$) *and checking the validity of* σ_{i_0} *as follows: for* $\mathbf{ID}_{i_j}$, $\mathbf{M}_{i_j}$, *and* σ_{i_j}, *we first divide* σ_{i_j} *into two parts; lower* k *bits,* σ'_{i_j}, *and the remaining part,* $\sigma_{M,i_{j-1}}$. *Then, we compute* $b||s_{i_j}||w_{i_j} = (\sigma'_{i_j})^{e_{i_j}} (\bmod n_{i_j})$, *where* $|b| = 1$, $|s_{i_j}| = k - k_1 - 1$, *and* $|w_{i_j}| = k_1$. *If* $b \neq 0$, *we reject the signature. Otherwise, we recover* $\sigma_{R,i_{j-1}}||r_{i_j} = s_{i_j} \oplus G(w_{i_j})$ *and* $\sigma_{L,i_{j-1}} = H(\mathbf{ID}_{i_j}, \mathbf{M}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}) \oplus w_{i_j}$, *where* $|\sigma_{R,i_{j-1}}| = k - k_0 - k_1 - 1$ *and* $|r_{i_j}| = k_0$. *We then divide* $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}}||ID_{i_j}$ *and* $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}}||m_{i_j}$, *and recover* $\sigma_{i_{j-1}} = \sigma_{L,i_{j-1}}||\sigma_{M,i_{j-1}}||\sigma_{R,i_{j-1}}$, *and execute the verification for* i_{j-1}*-th signer's signature.*
Finally, if $\sigma_{i_0} = 0^{k-k_0+k_1-1}$, *we accept the signature. Otherwise, we reject it.*

The increase of length of signature per signer in RSA-S-PSS-MSS is $k_0 + 1$ bits; which is the same as that in PFDH-MSS. Note that, by embedding a part of the message (m_0) into the recovery part of the first signer[7] ($\sigma_{i_0,R}$), we can reduce the bandwidth (the length of message and signature) compared to PFDH-MSS in a general case (see section 6). As we will see in section 5, S-PSS-MSS also has tight security.

4 Optimality for Probabilistic Multisignature Schemes

In this section, we give an optimal proof (*general result*) for multisignature scheme. The optimal proof gives an upper bound for reduction efficiency (the relation between the hardness of solving the mathematical problem and that of breaking the signature scheme); this relation derives the security parameter

[7] The verification algorithm checks if $\sigma_{i_0,L}||\sigma_{i_M,0}$ equals 0^{k-1-k_0}.

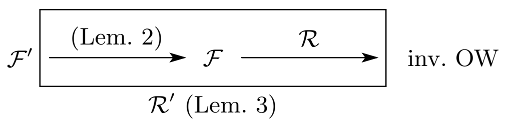


Fig. 2. Proof Strategy of Theorem 1

when the optimal "reduction" algorithm is utilized in the security proof. It is reasonable that the more efficient the reduction in the security proof is constructed, the shorter is the random salt sufficient for ensuring the security; namely, this relation gives the lower bound for the security parameter (especially, the length of random salt). We first review the definition of "reduction."

Definition 3 (Reduction, [4]). *We say that a reduction $\mathcal{R}$ $(t_R, q_\Sigma, q_H, \epsilon_F, \epsilon_R)$-reduces inverting a multiplicative (homomorphic) trapdoor permutation f (i.e., $f(ab) = f(a)f(b)$ holds for arbitrary a and b, e.g., the RSA function) to break a multisignature scheme if, upon input η and after running any forger that outputs the forgery in $(t_F, q_\Sigma, q_H, \epsilon_F)$, $\mathcal{R}$ outputs $f^{-1}(\eta)$ with probability more than ϵ_R, within an additional running time of t_R.*

Utilizing the "reduction" $\mathcal{R}$, we can derive the optimal proof for multisignature schemes ($\mathcal{MSS}$).

Theorem 1 (Optimal Proof for Probabilistic Multisignatures). *Let $\mathcal{F}$ be a forger who breaks $\mathcal{MSS}$ containing L signers in $(\tau_F, q_\Sigma, q_H, \epsilon)$ in accordance with EUF-ACMA &AIA, in the case of being allowed to make at most $L-1$ corrupt and masquerading queries in total. Let $\mathcal{R}$ be a reduction who reduces inverting a multiplicative trapdoor permutation f to break $\mathcal{MSS}$ in $(t_R, q_H, q_\Sigma, \epsilon_F, \epsilon_R)$. $\mathcal{R}$ can run or rewind $\mathcal{F}$ at most r times. From $\mathcal{R}$, we can construct an inverter $\mathcal{I}$ who breaks the trapdoor permutation. Here,*

$$\epsilon_I \geq \epsilon_R - r\epsilon_F \frac{2^{k_0+2}}{q_\Sigma}, \quad t_I \leq (r+1)t_R.$$

holds.

Suppose that the inverting problem is hard to solve ($\epsilon_I \approx 0$), that there exists an efficient reduction with $\epsilon_R = 1$, and that there exists a forger with constant (non-negligible) success probability (*e.g.*, $\epsilon_F = \frac{1}{4}$) with $r = 1$, then we have $\epsilon_I \geq 1 - 2^{k_0}/q_\Sigma$. If k_0 is smaller than $\log_2 q_\Sigma$, then ϵ_I is not negligible, and this contradicts the assumption of $\epsilon_I \approx 0$. Hence, we must choose k_0 more than $\log_2 q_\Sigma$. That is, Theorem 1 claims that the lower bound for the length of random salt is also $k_0 \approx \log_2 q_\Sigma$; which is the same as the case of an ordinary probabilistic signature scheme ($\mathcal{PS}$) [4]. Namely, our aim is to construct a provably secure probabilistic multisignature scheme in which the optimal length of the random salt is $k_0 \approx \log_2 q_\Sigma$ ($= 30$ bits).

We give the intuition of proof strategy of Theorem 1.

Coron [4] proved the optimal security of PSS by reducing a forger of PSS to one of (deterministic signature scheme) PSS0, and the reduction of PSS0 to that of PSS. In order to prove the optimal security of $\mathcal{MSS}$, we give the reduction between $\mathcal{MSS}$ and $\mathcal{PS}$. Figure 2 explains the situation: At first, we give the reduction from a forger $\mathcal{F}$ of $\mathcal{MSS}$ (*e.g.*, PFDH-MSS) to a forger $\mathcal{F}'$ of $\mathcal{PS}$ (resp. PFDH), and then, construct a reduction $\mathcal{R}'$ of $\mathcal{PS}$ from the forger $\mathcal{F}$ and a reduction $\mathcal{R}$. The relation (upper-bound) between $\mathcal{F}$ and $\mathcal{R}$, claimed in Theorem 1, comes from the optimal security of $\mathcal{PS}$ (PFDH and PSS in [4]). The proof is described in Appendix A.

5 Security Results and Optimal Length of Random Salt

This section first proves that the probabilistic multisignature schemes described in section 3 have tight security. The following subsection gives other security results for the schemes, which derive the upper bound for security parameter (the length of random salt), and find the optimal length of random salt in each scheme. For simplicity, we only discuss the cases of PFDH-MSS and S-PSS-MSS below. Almost the same discussion and results (tight security and optimal length of a random salt) hold for PSS-MSS.

5.1 Security Results

In this subsection, we give the security results of PFDH-MSS and S-PSS-MSS with multiplicative trapdoor permutation f including the RSA function. The following results show that these schemes have tight security. Note that even if we remove the assumption of multiplicative, the security of these schemes can be ensured.

Theorem 2 (Security of PFDH-MSS). *Let $\mathcal{F}$ be a forger who breaks PFDH-MSS containing L signers in $(\tau, q_\Sigma, q_H, \epsilon)$ in accordance with EUF-ACMA&AIA, in the case of being allowed to make at most $L-1$ corrupt and masquerading queries in total. Then we can break the one-wayness of f within time bound τ' and with success probability $\mathsf{Succ}^{\mathsf{ow}}(\tau')$:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geq \frac{1}{L}(\epsilon - \frac{q_\Sigma(q_H+q_\Sigma)}{2^{k_0}} - \frac{q_\Sigma+1}{2^{k-1}} - \frac{q_H+q_\Sigma}{2^N}) \\ \tau' \leq \tau + ((q_H+q_\Sigma)+L)\mathsf{T}_f \end{cases}$$

where T_f denotes the time complexity of f and N is a constant[8].

Theorem 3 (Security of S-PSS-MSS). *Let $\mathcal{F}$ be a forger who breaks S-PSS-MSS containing L signers in $(\tau, q_\Sigma, q_G, q_H, \epsilon)$ in accordance with EUF-ACMA&AIA, in the case of being allowed to make at most $L-1$ corrupt*

[8] In simulation in the security proof, we should find z_{i_j} (signature) such that the MSB of $z_{i_j}^{e_{i_j}} \bmod n_{i_j}$ equals 0. If we choose z_{i_j} at random and the MSB is 1 for N times, we abort the proof.

and masquerading queries in total. Then we can break the one-wayness of f within time bound τ' and with success probability $\mathsf{Succ}^{\mathsf{ow}}(\tau')$*:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geq \frac{1}{L}(\epsilon - \frac{q_H q_\Sigma}{2^{k_0}} - \frac{q_H((q_H+q_\Sigma)^2+q_G)+q_\Sigma(q_G+q_H+q_\Sigma)+1}{2^{k_1}} - \frac{q_H+q_\Sigma}{2^N}) \\ \tau' \leq \tau + ((q_H + q_\Sigma)N + L)T_f \end{cases}$$

where T_f denotes the time complexity of f and N is a constant.

We will give the proofs of these theorems in the full version of this paper. The strategy of these theorems is almost the same as the proof of Theorem 5 given in Appendix B.

5.2 Estimation of Optimal Length of Random Salt

Coron introduces the notion of "reduction" for two purposes: to prove that FDH does not have tight security and to estimate the optimal length of a random salt in PSS. As we described in section 4, with respect to probabilistic multisignature schemes, the lower bound for the length of random salt can be estimated, $k_0 \approx \log_2 q_\Sigma$. The optimal proof discussed in section 4 does not ensure the security of the schemes at all (namely, the optimal proof only means that if the scheme is secure and there are reduction algorithms to prove the security, we can evaluate the upper bound for reduction efficiency); however, we have already shown that the security of the schemes (PFDH-MSS and S-PSS-MSS) can be proven. This now raises the question of how long a random salt is sufficient (the upper bound for the length of a random salt) in order to ensure the security. Following the technique of Coron [4], we obtain another security result for PFDH-MSS and S-PSS-MSS, respectively.

Theorem 4 (Another security result for PFDH-MSS). *Let $\mathcal{F}$ be a forger who breaks PFDH-MSS containing L signers in $(\tau, q_\Sigma, q_H, \epsilon)$ in accordance with EUF-ACMA &AIA, in the case of being allowed to make at most $L-1$ corrupt and masquerading queries in total. Then we can break the one-wayness of f within time bound τ' and with success probability* $\mathsf{Succ}^{\mathsf{ow}}(\tau')$*:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geq \frac{1}{L}\left(1 + \frac{6q_\Sigma}{2^{k_0}}\right)^{-1}\left(\epsilon - \frac{q_\Sigma+1}{2^{k-1}} - \frac{q_H+q_\Sigma}{2^N}\right) \\ \tau' \leq \tau + ((q_H + q_\Sigma)N + L)T_f \end{cases}$$

where T_f denotes the time complexity of f and N is a constant.

Theorem 5 (Another security result for S-PSS-MSS). *Let $\mathcal{F}$ be a forger who breaks S-PSS-MSS containing L signers in $(\tau, q_\Sigma, q_G, q_H, \epsilon)$ in accordance with EUF-ACMA &AIA, in the case of being allowed to make at most $L-1$ corrupt and masquerading queries in total. Then we can break the one-wayness of f within time bound τ' and with success probability* $\mathsf{Succ}^{\mathsf{ow}}(\tau')$*:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geq \frac{1}{L}\left(1 + \frac{6q_\Sigma}{2^{k_0}}\right)^{-1}\left(\epsilon - \frac{q_H((q_H+q_\Sigma)^2+q_G)+q_\Sigma(q_G+q_H+q_\Sigma+1)+1}{2^{k_1}} - \frac{q_H+q_\Sigma}{2^N}\right) \\ \tau' \leq \tau + ((q_H + q_\Sigma)N + L)T_f \end{cases}$$

where T_f denotes the time complexity of f and N is a constant.

The proof follows the technique of that of PSS (Theorem 4 in [4]). When a new message is queried to a hash or signing oracle, we construct a list of (at most q_Σ) random salts with probability β. We do not embed η (see the Appendix B) for the message and salt if the salt belongs to the list, and embed η otherwise. Note that we must estimate the optimal β carefully because, if k_0 is small ($\approx \log_2 q_\Sigma$) and β is large (≈ 1), there are few random salts for which we embed η; which decrease the success probability of inverting a trapdoor permutation. We give the proof of Theorem 5 in Appendix B; Theorem 4 is proven from almost the same (but simpler) discussion as the proof of Theorem 5.

Now, let us discuss the optimal length of a random salt in these schemes in detail. With regard to PFDH-MSS, Theorem 4 claims that even if we enlarge the length of a random salt k_0 more than $\log_2 q_\Sigma (\approx 30)$, the security does not improve, from the same discussion as in PSS [4]. On the other hand, as we noted, Theorem 1 confirms that at least $k_0 \approx \log_2 q_\Sigma$ bits are required in order to ensure the security, even if we construct the optimal (the best) reduction algorithm for the security proof. From these two results, we conclude that $k_0 \approx \log_2 q_\Sigma$ is optimal in PFDH-MSS. The same consideration can be made for the case of S-PSS-MSS, *i.e.*, in S-PSS-MSS, we also know that $k_0 \approx \log_2 q_\Sigma$ is optimal. This is not only a theoretical impact. PFDH-MSS and S-PSS-MSS realize the optimal (and minimum) length of a random salt and are promising schemes in practical use.

6 Discussion

This section compares the multisignature schemes described in section 3 with the original sequential aggregate signature scheme proposed by Lysyanskaya et al. [8]. The original sequential aggregate signature scheme is constructed from FDH; the signing and verification algorithms are handled by removing the random salt (r_{i_j} and $\mathbf{r}_{i_j}$) from the protocol of PFDH-MSS (Protocol 1). We call the original FDH based sequential aggregate signature scheme FDH-MSS.

With regard to security tightness, the security of FDH-MSS is not tight enough; on the other hand, as we discussed, the probabilistic multisignature schemes have tight security. It is a well-known fact that, in ordinary signature schemes (signed by one signer), we can construct signature schemes which realize tight security. Our result indicates that this principle holds for multiple user setting, multisignature schemes.

We now proceed to discuss the length of signature in probabilistic multisignature schemes. Let us consider the signature signed by L' signers. The length of signature signed with PFDH-MSS and PSS-MSS (utilized in the RSA permutation) is clearly $k + (L' - 1)k_0$ and $k + (L' - 1)(k_0 + k_1 + 1)$ bits, respectively. In respect of S-PSS-MSS, in order to ensure the security and realize the simple construction[9], we set σ_{i_0} by $k - k_0 + k_1 - 1$ bits "0". Through careful consider-

[9] In our proof, it is essential $\sigma_{i_0} = 0^{k-k_0+k_1-1}$. If another proof technique is invented, the length of σ_{i_0} may be decreased; however, if the length of σ_{i_0} is less than $k - k_0 - 1$ bits, the signing algorithm of the first ordered signer should be described apart from that of other signers.

ation, we can estimate the signature length by $k + (L' - 1)(k_0 + 1) + k_1$ bits in S-PSS-MSS.

The length of signature signed with S-PSS-MSS is greater than that with PFDH-MSS by k_1 bits; however, note that it does not depend on the number of signers (constant). Moreover, by embedding a part of the message into the recovery part of the first signer, σ_{R,i_0}, we can reduce the bandwidth (the length of message and signature) compared to PFDH-MSS if $k_1 < k - k_1 - k_0 - 1$. For example, if we utilize the RSA function and SHA-256; $k = 1024$, $k_1 = 256$, $k_0 = 30$ (assume $q_\Sigma = 2^{30}$) are recommended, and the above inequality holds.

The increases of signature size in PFDH-MSS and S-PSS-MSS are $(k_0 + 1)$ bits, respectively. From the fact that we should utilize a random salt in order to realize tight security and that the random salt should be attached with or recovered from a signature since it is required in the verification step, it is inevitable that the length of signature is enlarged by more than or equal to k_0 bits per signer. To sum up, the increases of signature size in PFDH-MSS and S-PSS-MSS are optimal.

7 Conclusion

This paper first proved that PFDH-MSS, PSS-MSS, and S-PSS-MSS have tight security. Second, we gave the optimal proof (general result) for probabilistic multisignature schemes, which derives the lower bound for length of random salt. Third, we proved the schemes' security in the other direction, which gives the upper bound for the length of random salt, and confirmed that these bounds are (almost) identical and give the optimal length of a random salt; the optimal length of a random salt utilized in probabilistic multisignature schemes is the same as that in ordinary signature schemes. Two of the schemes, PFDH-MSS and S-PSS-MSS, are promising schemes in terms of security tightness and optimal signature length.

References

1. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
2. M. Bellare and P. Rogaway. The exact security of digital signatures –how to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Berlin, Heidelberg, New York, 1996. Springer-Verlag.
3. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiability encrypted signature form bilinear maps. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, New York, 2003. Springer-Verlag.
4. J. S. Coron. Optimal security proofs for PSS and other signature schemes. In L. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287, Berlin, Heidelberg, New York, 2002. Springer-Verlag.

5. K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development, (71)*, pages 1–8, 1983.
6. K. Kawauchi and M. Tada. On the exact security of multisignature schemes based on RSA. In R. S. Naini and J. Seberry, editors, *The Eighth Australasian Conference on Information Security and Privacy (ACISP 2003)*, volume 2727 of *Lecture Notes in Computer Science*, pages 336–349, Berlin, Heidelberg, New York, 2003. Springer-Verlag.
7. K. Kawauchi and M. Tada. On the security and the efficiency of multi-signature schemes based on a trapdoor one-way permutation. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E88–A(5), 2005.
8. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90, Berlin, Heidelberg, New York, 2004. Springer-Verlag.
9. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *CCS'01, Eighth ACM Conference on Computer and Communications Security*, 2001.
10. S. Mitomi and A. Miyaji. A general model of multisignature schemes with message flexibility, order flexibility, and order verifiability. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84–A(10):2488–2499, 2001.
11. 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.

A Proof of Theorem 1

We first review the following lemma, which ensures the optimal length of a random salt in (general) probabilistic signature schemes, $\mathcal{PS}$.

Lemma 1 (Optimality for the random salt in $\mathcal{PS}$, [4]). Let $\mathcal{F}'$ be a forger who breaks $\mathcal{PS}$ in $(\tau'_F, q'_\Sigma, q'_G, q'_H, \epsilon'_F)$ in accordance with EUF-ACMA, and let $\mathcal{R}'$ be a reduction who reduces inverting a multiplicative trapdoor permutation f to break PSS in $(t'_R, q'_G, q'_H, q'_\Sigma, \epsilon'_F, \epsilon'_R)$. $\mathcal{R}'$ can run or rewind $\mathcal{F}'$ at most r times. From $\mathcal{R}'$, we can construct an inverter $\mathcal{I}'$ who breaks the trapdoor permutation within time bound t'_I with probability more than ϵ'_I. Here,

$$\epsilon'_I \geq \epsilon'_R - r\epsilon'_F \frac{2^{k_0+2}}{q'_\Sigma}, \quad t'_I \leq (r+1)t'_R.$$

Let $\mathcal{F}'$ be a forger against $\mathcal{PS}$ and $\mathcal{R}'$ a reduction from inverting the trapdoor permutation to breaking $\mathcal{PS}$. The following lemmas give relation between $\mathcal{F}$ and $\mathcal{F}'$, and between $\mathcal{R}$ and $\mathcal{R}'$, respectively. Lemma 2 gives the description of a reduction from $\mathcal{F}$ to $\mathcal{F}'$ and the relation of their success probability, etc., and Lemma 3 gives the reduction from $\mathcal{R}'$ to $\mathcal{R}$ and the relation of their success probability, etc. (Figure 2). Lemma 1 gives the relation (upper bound) between ϵ'_F and ϵ'_R; and from which we derive an upper bound of ϵ_R with respect to ϵ_F.

Lemma 2 (Relation between $\mathcal{F}$ and $\mathcal{F}'$). Let $\mathcal{F}'$ be a forger who breaks $\mathcal{PS}$ in accordance with $\mathsf{EUF\text{-}ACMA}$ in $(t'_F, q'_\Sigma, q'_G, q'_H, \epsilon'_F)$. Then, from $\mathcal{F}'$, we can construct a forger who breaks $\mathcal{MSS}$ containing L signers in accordance with $\mathsf{EUF\text{-}ACMA\&AIA}$ in $(t_F, q_\Sigma, q_G, q_H, \epsilon_F)$, where

$$t_F = t'_F, \quad q_G = q'_G, \quad q_H = q'_H, \quad q_\Sigma = q'_\Sigma, \quad \epsilon_F = \epsilon'_F.$$

(Proof of Lemma 2) We construct $\mathcal{F}$ who breaks $\mathcal{MSS}$, utilizing $\mathcal{F}'$ who breaks $\mathcal{PS}$ as an oracle. $\mathcal{F}$, given a set of public keys $(\mathsf{pk}_1, \cdots, \mathsf{pk}_L)$, outputs a forgery after making signing queries[10] to signing oracles. $\mathcal{F}$ first chooses $T \overset{R}{\leftarrow} [1, L]$ and inputs pk_T to $\mathcal{F}'$ as a key of a signer in $\mathcal{PS}$.

After receiving the key, $\mathcal{F}'$ makes hash and signing queries, and finally, outputs a forgery corresponding to the key in $\mathcal{PS}$. When $\mathcal{F}$ receives the query made by $\mathcal{F}'$, $\mathcal{F}$ transfers the queries to the corresponding oracle, receives an answer and returns the answer to $\mathcal{F}'$. Finally, $\mathcal{F}'$ outputs a forgery (of $\mathcal{PS}$), and then, $\mathcal{F}$ outputs it as a forgery[11] in $\mathcal{MSS}$. Since the first ordered signer of $\mathcal{MSS}$ behaves in the same manner as in $\mathcal{PS}$ (see Definition 1, etc.,), $\mathcal{F}$ wins the game and Lemma 2 holds.

Lemma 3 (Relation between $\mathcal{R}'$ and $\mathcal{R}$). Let $\mathcal{R}$ be a reduction who utilizes $\mathcal{F}$ to reduce inverting the trapdoor permutation to break $\mathcal{MSS}$ in $(t_R, q_\Sigma, q_G, q_H, \epsilon_F, \epsilon_R)$. Then, we can construct, by utilizing $\mathcal{R}$ as an oracle, a reduction $\mathcal{R}'$ who reduces inverting the trapdoor permutation to break $\mathcal{PS}$ in $(t'_R, q'_\Sigma, q'_G, q'_H, \epsilon'_F, \epsilon'_R)$, where

$$t_R = t'_R, \quad q_G = q'_G, \quad q_H = q'_H, \quad q_\Sigma = q'_\Sigma, \quad \epsilon_F = \epsilon'_F, \quad \epsilon_R = \epsilon'_R.$$

(Proof of Lemma 3) We construct $\mathcal{R}'$ which utilizes $\mathcal{F}'$ (a forger against $\mathcal{PS}$) to reduce inverting the trapdoor permutation to break $\mathcal{PS}$. From Lemma 2, we can construct a forger $\mathcal{F}$ who breaks $\mathcal{MSS}$ from a forger F' for $\mathcal{PS}$. Then, from $\mathcal{F}$ using $\mathcal{R}$, we can invert the one-way permutation. Therefore, from $\mathcal{R}$, we can construct a reduction $\mathcal{R}'$ and Lemma 3 holds.

By substituting the results of Lemmas 2 and 3 into Lemma 1, we have the relation described in Theorem 1.

B Proof of Theorem 5

In this section, we prove Theorem 5, following the approach for PSS of Coron (general case in proofs of Theorems 2 and 4 in [4]). The proof of Theorem 4 is derived from the slight modification of (indeed, the proof is easier than)

[10] In addition, $\mathcal{F}$ also outputs corrupt and masquerading queries, but, for simplicity, we only describe the action for the signing query.

[11] $\mathcal{F}$ can query the forgery to other signing oracle (Σ_{i_j}, $i_j \neq T$) to acquire a multisignature signed by multiple signers. Note that the forgery, output by $\mathcal{F}'$, has not been queried to Σ_T by $\mathcal{F}'$ nor $\mathcal{F}$.

the following proof. Hereafter, we denote the RSA function (computation $x^{e_{i_j}}$ mod n_{i_j} for x) by $f_{i_j}(x)$, and its inverse (resp. $y^{d_{i_j}}$ mod n_{i_j} for y) by $f_{i_j}^{-1}(y)$.

We give a security proof by conducting a contradiction; namely, if there is a forger $\mathcal{F}$ against S-PSS-MSS, then we can construct an inverter $\mathcal{I}$ who can invert a trapdoor one-way permutation (*e.g.*, solving the RSA problem) by using $\mathcal{F}$ as an oracle with non-negligible success probability. The following subsection assumes that there exists a forger $\mathcal{F}$ who breaks S-PSS-MSS with $(\epsilon, q_\Sigma, q_H, t)$, and constructs the inverter $\mathcal{I}$.

B.1 Construction of Inverter $\mathcal{I}$

We give the construction of inverter $\mathcal{I}$ that breaks the *one-wayness* of multiplicative permutation f on η, by using forger $\mathcal{F}$ that $(\tau, q_\Sigma, q_G, q_H, \epsilon)$-breaks $\mathcal{MSS}$ containing L signers in accordance with $\mathsf{EUF\text{-}ACMA\&AIA}$, as follows: we first guess the target signer[12] P_T at random, and then, we input public key $\mathsf{pk}_T = f$ to $\mathcal{F}$, answers the queries that $\mathcal{F}$ asks to the random oracles, and to the signing oracle. Finally, we receive forgery $\mathbf{ID}^*_{i_{L'}}, \mathbf{M}^*_{i_{L'}}, \sigma^*_{i_{L'}}$ (or stop $\mathcal{F}$ after its running time τ is over).

In simulating random oracles G and H, we construct input/output lists, $\mathsf{G\text{-}List}$ and $\mathsf{H\text{-}List}$, respectively. $\mathsf{G\text{-}List}$ holds $(w_{i_j}, G(w_{i_j}))$, the pairing of query w_{i_j} and answer $G(w_{i_j})$. $\mathsf{H\text{-}List}$ holds $(b, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j})$, the tuple of bit[13] $b \in \{-1, 0, 1\}$, first part[14] of the previous signer's signature $\sigma_{L,i_{j-1}}$, query $\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}$, $\sigma_{M,i_{j-1}}$, $\sigma_{R,i_{j-1}}$, and r_{i_j}, guarantee z_{i_j} for signing queries, and answer w'_{i_j}.

Moreover, we construct list L_i of at most q_Σ random salts in $\{0,1\}^{k_0}$. We first fix probability $\beta \in [0,1]$ and set $i = 0$. When a message $\mathbf{M}_{i_j}$ appears for the first time in a H or signing query with respect to ID_T, we increment i and set $\mathbf{M}^{(i)} = \mathbf{M}_{i_j}$. Then, with probability β, we generate and add a random salt in $\{0,1\}^{k_0}$ into list L_i. If the other case (with probability $1-\beta$) or if L_i contains q_Σ elements, we stop adding one to the list. Roughly speaking, we utilize the salt in order to simulate the answers to signing queries (without embedding η).

In this strategy, in order for $\mathcal{F}$ to output valid signature, $\mathbf{ID}_{i_{L'}}$, $\mathbf{M}^*_{i_{L'}}$, and $\sigma^*_{L'}$, $\mathcal{F}$ queries H on $\mathbf{M}^*_{i_j}$, $\mathbf{ID}^*_{i_j}$, $\sigma^*_{M,i_{j-1}}$, $\sigma^*_{R,i_{j-1}}$, and $r^*_{i_j}$, where $i_j = T$; in this case, if $r^*_{i_j}$ is not in list L_i, we can find[15] $f^{-1}(\eta) = \frac{[\sigma^*_{i_j}]_k}{z^*_{i_j}}$.

[12] From the definition of a forgery (Definition 2), there is at least one signer (*the target signer*), who is not corrupted by forger $\mathcal{F}$ (nor whom $\mathcal{F}$ masquerades as) and is not requested to sign on the forged message, and whose signature is in the forgery.

[13] $b = 1$ means that η is embedded for the query $m, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}$, and $b = 0$ means that η is not implanted for the query. Moreover, $b = -1$ means that we answer a random string in the case where $m, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}$ should be invalid signature for all $\sigma_{L,i_{j-1}}$.

[14] This $\sigma_{L,i_{j-1}}$ is not included in the hash query; we recover it by locating the list.

[15] Here, $[a]^b$ denotes the b most significant bits of a, while $[a]_b$ denotes the b least significant bits of a.

Hereafter, we assume that $\mathcal{F}$ does not (hash-)query about the same message. Furthermore, we assume that $\mathcal{F}$ attacks one signer[16] in $\mathcal{G}$.

Initialization and answering the corrupt and masquerading queries
$\mathcal{I}$ runs the key generation algorithm for P_{i_j} $(j = 0, \cdots L)$ except for P_T. If $\mathcal{F}$ outputs the corrupt or masquerading query to P_T, we abort (fail to simulate). For the corrupt query to P_{i_j} $(i_j \neq T)$, $\mathcal{I}$ answers pk_{i_j} to $\mathcal{F}$. For the masquerading query to P_{i_j} $(i_j \neq T)$, $\mathcal{I}$ receives pk_{i_j} from $\mathcal{F}$ and registers it.

Answering the random oracle queries to G
For query w_{i_j} to G, we first locate $(w_{i_j}, G(w_{i_j})) \in$ G-List; if it exists, we answer $G(w_{i_j})$. Otherwise, we choose random string from $\{0,1\}^{k-k_0-1}$, considers it as $G(w_{i_j})$, answer to $\mathcal{F}$, and add $(w_{i_j}, G(w_{i_j}))$ to G-List.

Answering the random oracle queries to H
For query $\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}$, and r_{i_j} to H, we first locate $(*, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, *, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, *, w') \in$ H-List; if it exists, we answer w'. Otherwise, we simulate an answer in the following way (five cases).
(Case 1): $i_j = T,\ j = 1$: At first, we check if $\sigma_{M,i_{j-1}}||\sigma_{R,i_{j-1}} = 0^{k-k_1-1}$; if not, then we choose $w'_{i_j} \overset{R}{\leftarrow} \{0,1\}^{k_1}$, set $(-1, \mathbf{M}_{i_j}, ID_{i_j}, \phi, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, \phi, w'_{i_j}) \rightarrow$ H-List, and return w'_{i_j}.

Otherwise, we set $b' = 0$ if $r_{i_j} \in L_i$ or $b' = 1$ otherwise. Then, we repeatedly choose $z_{i_j} \overset{R}{\leftarrow} \{0,1\}^k$ until we have $f_{i_j}(z_{i_j})\eta^{b'} = 0||s_{i_j}||w_{i_j}$, where $|s_{i_j}| = k-k_1-1$ and $|w_{i_j}| = k_1$. If we cannot find such z_{i_j} through N times trial, we abort. Otherwise, we set $\sigma_{L,i_{j-1}} = 0^{k_1}$. If $(w_{i_j}, *) \in$ G-List, then we abort. Otherwise, we simulate $G(w_{i_j}) = s_{i_j} \oplus (\sigma_{R,i_{j-1}}||r_{i_j})$ by preserving $(w_{i_j}, G(w_{i_j}))$ in G-List, and simulate $w'_{i_j} = H(\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}) = w_{i_j} \oplus 0^{k_1} = w_{i_j}$ by adding $(b', \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j})$ to H-List. Finally, we answer w'_{i_j}.
(Case 2): $i_j = T,\ j \geq 2$: In this case, we first count the number of elements $(b, \mathbf{M}_{i_{j-1}}, \mathbf{ID}_{i_{j-1}}, *, \sigma_{M,i_{j-2}}, *, *, z_{i_{j-1}}, *) \in$ H-List such that $[\sigma_{M,i_{j-2}}||z_{i_{j-1}}]_{|(\sigma_{M,i_{j-1}}||}$

If there is one element, satisfying both the above condition and $b = 0$, *i.e.*, if $(0, \mathbf{M}_{i_{j-1}}, \mathbf{ID}_{i_{j-1}}, \sigma_{L,i_{j-2}}, \sigma_{M,i_{j-2}}, \sigma_{R,i_{j-2}}, r_{i_{j-1}}, z_{i_{j-1}}, w_{i_{j-1}}) \in$ H-List, we set $b' = 0$ if $r_{i_j} \in L_i$ or $b' = 1$ otherwise. Then, we repeatedly choose $z_{i_j} \overset{R}{\leftarrow} \{0,1\}^k$, until we have $f_{i_j}(z_{i_j})\eta^{b'} = 0||s_{i_j}||w_{i_j}$, where $|s_{i_j}| = k - l_1 - 1$ and $|w_{i_j}| = k_1$. If we cannot find such z_{i_j} through N times trial, we abort. Otherwise, we set $\sigma_{L,i_{j-1}} = [\sigma_{M,i_{j-2}}||z_{i_{j-1}}]^{k_1}$. If $(w_{i_j}, *) \in$ G-List, then we abort. Otherwise, we simulate $G(w_{i_j}) = s_{i_j} \oplus (\sigma_{R,i_{j-1}}||r_{i_j})$ by preserving $(w_{i_j}, G(w_{i_j}))$ in

[16] Generally, there may be more than one *target signer*, *e.g.*, $P_{T_1}, \cdots, P_{T_l}$. We choose $t \overset{R}{\leftarrow} [1, l]$ and regard P_t as the target; this procedure does not affect the behavior of $\mathcal{F}$. Therefore, we can estimate the probability that $\mathcal{I}$ can guess the target signer correctly by exactly $\frac{1}{L}$.

G-List, and simulate $w'_{i_j} = H(\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}) = w_{i_j} \oplus \sigma_{L,i_{j-1}}$ by adding $(b', \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j})$ to H-List. Finally, we answer w'_{i_j}.

If there is more than one element, satisfying the above condition, such that $b = 0$, then we abort. The following gives the reason why we have to abort in this case; assume there are two elements, *el1* $= (\mathbf{M}_{i_{j-1}}, \mathbf{ID}_{i_{j-1}}, \sigma_{M,i_{j-2}}, \sigma_{R,i_{j-2}}, z_{i_{j-1}})$ and *el2* $= (\widetilde{\mathbf{M}}_{i_{j-1}}, \widetilde{\mathbf{ID}}_{i_{j-1}}, \widetilde{\sigma}_{M,i_{j-2}}, \widetilde{\sigma}_{R,i_{j-2}}, \widetilde{z}_{i_{j-1}})$, and assume that we embed η for *el1* to simulate the answer w'_{i_j}. In this case, if $\mathcal{F}$ outputs a forgery corresponding to *el1*, then we can compute $f^{-1}(\eta)$ as we will see in this proof; however, if $\mathcal{F}$ outputs a forgery corresponding to *el2* using w'_{i_j}, then we cannot invert f even if the forgery is valid.

Otherwise, *i.e.*, if there is no element or if there is one more element(s), satisfying the above condition with $b = -1$, then we choose $w'_{i_j} \overset{R}{\leftarrow} \{0,1\}^{k_1}$, set $(-1, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \phi, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, \phi, w'_{i_j}) \to$ H-List, and return w'_{i_j}.

(Case 3): $i_j \neq T$, $j = 1$: Almost the same as (Case 1), without embedding η.

(Case 4): $i_j \neq T$, $j \leq 2$, $i_{j-1} \neq T$: Almost the same as (Case 2), without embedding η.

(Case 5): $i_j \neq T$, $j \leq 2$, $i_{j-1} = T$: In this case, we first count the number of elements $(b, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, *, \sigma_{M,i_{j-2}}, *, *, z_{i_{j-1}}, *) \in$ H-List which satisfies the following condition; $[\sigma_{M,i_{j-2}} || z_{i_{j-1}}]_{|(\sigma_{M,i_{j-1}}||\sigma_{R,i_{j-1}})|} = \sigma_{M,i_{j-1}} || \sigma_{R,i_{j-1}}$.

If there is one element, satisfying both the above condition and $b = 0$, *i.e.*, if $(0, \mathbf{M}_{i_{j-1}}, \mathbf{ID}_{i_{j-1}}, \sigma_{L,i_{j-2}}, \sigma_{M,i_{j-2}}, \sigma_{R,i_{j-2}}, r_{i_{j-1}}, z_{i_{j-1}}, w'_{i_{j-1}}) \in$ H-List, then we repeatedly choose $z_{i_j} \overset{R}{\leftarrow} \{0,1\}^k$, until we have $f_{i_j}(z_{i_j}) = 0 || s_{i_j} || w_{i_j}$, where $|s_{i_j}| = k - k_1 - 1$ and $|w_{i_j}| = k_1$. If we cannot find such z_{i_j} through N times trial, we abort. Otherwise, we set $\sigma_{L,i_{j-1}} = [\sigma_{M,i_{j-2}} || z_{i_{j-1}}]^{k_1}$. If $(w_{i_j}, *) \in$ G-List, then we abort. Otherwise, we simulate $G(w_{i_j}) = s_{i_j} \oplus (\sigma_{R,i_{j-1}} || r_{i_j})$ by preserving $(w_{i_j}, G(w_{i_j}))$ in G-List, and simulate $w'_{i_j} = H(\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}) = w_{i_j} \oplus \sigma_{L,i_{j-1}}$ by adding $(0, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j})$ to H-List. Finally, we answer w'_{i_j}.

If there is more than one element, satisfying the above condition, such that $b = 0$, then we abort.

If there is one or more elements, satisfying both the above condition and $b = 1$, *i.e.*, if $(1, \mathbf{M}_{i_{j-1}}, \mathbf{ID}_{i_{j-1}}, \sigma_{L,i_{j-2}}, \sigma_{M,i_{j-2}}, \sigma_{R,i_{j-2}}, r_{i_{j-1}}, z_{i_{j-1}}, w'_{i_{j-1}}) \in$ H-List, then we check if $f([\sigma_{M,i_{j-1}} || \sigma_{R,i_{j-1}}]_k) = 0 || s_{i_{j-1}} || w_{i_{j-1}}$ holds; if yes, then $\mathcal{I}$ can compute $f^{-1}(\eta) = [\sigma_{M,i_{j-1}} || \sigma_{R,i_{j-1}}]_k / z_{i_{j-1}}$. If, for all tuple, the equality does not hold, then we choose $w'_{i_j} \overset{R}{\leftarrow} \{0,1\}^{k_1}$, set $(-1, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \phi, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, \phi, w'_{i_j}) \to$ H-List, and return w'_{i_j}.

Otherwise, *i.e.*, if there is no element or if there is one more element(s), satisfying the above condition, with $b = -1$, then we choose $w'_{i_j} \overset{R}{\leftarrow} \{0,1\}^{k_1}$, set $(-1, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \phi, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, \phi, w'_{i_j}) \to$ H-List, and return w'_{i_j}.

Answering the signing queries to Σ_{i_j} $(i_j = T)$
For query $\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}$ to Σ_{i_j}, we first check if there is at least one element in list L_i; if not, we abort. Otherwise, we choose $r_{i_j} \leftarrow L_i$ and $L_i = L_i \setminus \{r_{i_j}\}$, and then, locate $(0, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j}) \in$ H-List; if it exists, we answer $\sigma_{M,i_{j-1}}||z_{i_j}$.

Otherwise, we check if there exist $(-1, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j})$ in H-List; if yes, we answer $\sigma_{i_{j-1}}$ is invalid. Otherwise, we repeatedly choose $z_{i_j} \overset{R}{\leftarrow} \{0,1\}^k$, until we have $f_{i_j}(z_{i_j}) = 0||s_{i_j}||w_{i_j}$, where $|s_{i_j}| = k - k_1 - 1$ and $|w_{i_j}| = k_1$. If we cannot find such z_{i_j} through N times trial, we abort. Then, we check if $(w_{i_j}, *) \in$ G-List; if yes, we abort. Otherwise, we simulate $G(w_{i_j}) = s_{i_j} \oplus (\sigma_{R,i_{j-1}}||r_{i_j})$ by preserving $(w_{i_j}, G(w_{i_j}))$ in G-List, and simulate $w'_{i_j} = H(\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}) = w_{i_j} \oplus \sigma_{L,i_{j-1}}$ by adding $(0, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j})$ to H-List. Finally, we answer $\sigma_{M,i_{j-1}}||z_{i_j}$.

Answering the signing queries to Σ_{i_j} $(i_j \neq T)$
For query $\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}$ to Σ_{i_j}, we first choose $r_{i_j} \overset{R}{\leftarrow} \{0,1\}^{k_0}$, and then, locate $(0, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j}) \in$ H-List; if it exists, we answer $\sigma_{M,i_{j-1}}||z_{i_j}$.

Otherwise, we check if there exist $(-1, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j})$ in H-List; if yes, we answer $\sigma_{i_{j-1}}$ is invalid. Otherwise, we repeatedly choose $z_{i_j} \overset{R}{\leftarrow} \{0,1\}^k$, until we have $f_{i_j}(z_{i_j}) = 0||s_{i_j}||w_{i_j}$, where $|s_{i_j}| = k - k_1 - 1$ and $|w_{i_j}| = k_1$. If we cannot find such z_{i_j} through N times trial, we abort. Otherwise, we check if $(w_{i_j}, *) \in$ G-List; if yes, we abort. Otherwise, we simulate $G(w_{i_j}) = s_{i_j} \oplus (\sigma_{R,i_{j-1}}||r_{i_j})$ by preserving $(w_{i_j}, G(w_{i_j}))$ in G-List, and simulate $w'_{i_j} = H(\mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}) = w_{i_j} \oplus \sigma_{L,i_{j-1}}$ by adding $(0, \mathbf{M}_{i_j}, \mathbf{ID}_{i_j}, \sigma_{L,i_{j-1}}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}, z_{i_j}, w'_{i_j})$ to H-List. Finally, we answer $\sigma_{M,i_{j-1}}||z_{i_j}$.

Analysis (sketch) Let $\mathbf{M}^*_{i_{L'}}, \sigma^*_{i_{L'}}$ be a forgery output by $\mathcal{F}$; $\mathbf{ID}^*_{i_j}$, $s^*_{i_j}$s, $w'^*_{i_j}$s, $w^*_{i_j}$s, and $r^*_{i_j}$s are the corresponding elements. Note that, for $i_j = T$, if $(\mathbf{M}^*_{i_j}, \mathbf{ID}^*_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r^*_{i_j})$ is queried to H and η is embedded for the query (*i.e.*, $r_{i_j} \notin L_i$), then $\mathcal{I}$ can invert f for η. Let δ_1 and δ_2 be the probability of the failure in simulating an answer to H and signing queries, respectively. From the above notations, we have the following in the same manner as PSS (proof of Theorem 4 in [4]):

$$\mathsf{Succ}^{\mathsf{ow}}(\tau') \geq \frac{1}{L}(\epsilon - \delta_1 - \delta_2 - \frac{1}{2^{k_1}})\beta^{q_\Sigma}\left\{\sum_{j=0}^{q_\Sigma} \Pr[\sharp L_i = j](1 - \frac{1}{2^{k_0}})^j\right\} \quad (1)$$

Here, $\frac{1}{L}$ indicates the probability that $\mathcal{I}$ succeeds in guessing target signer P_T, and $\frac{1}{2^{k_1}}$ corresponds to the event in which $\mathcal{F}$ succeeds in outputting a forgery without querying the above hash query.

δ_1 is evaluated by $q_H(\frac{(q_H+q_\Sigma)^2+q_G}{2^{k_1}} + \frac{1}{2^N})$. This is because the probability of a failure in simulation of H (Case 1 to Case 5) is bounded by $\frac{(q_H+q_\Sigma)^2+q_G}{2^{k_1}} + \frac{1}{2^N}$;

for example in Case 2, the probability of $(w_{i_j}, *)$ being in $\mathsf{G\text{-}List}$ with universal distributed w_{i_j} is estimated by $\frac{q_G+q_H+q_\Sigma}{2^{k_1}}$, and the probability that there is more than one element, or that there is more than one valid multisignature which has the same lower bits is bounded by $\frac{(q_H+q_\Sigma)(q_H+q_\Sigma-1)}{2^{k_1}}$ from the discussion of birthday paradox. With regard to finding z_{i_j}, with probability $\frac{1}{2^N}$, we cannot find z_{i_j} such that the MSB of $f_{i_j}(z_{i_j})$ or $f_{i_j}(z_{i_j})\eta^{b'}$ equals 0. Therefore, we can estimate $\Pr[\mathsf{H'Bad}|\mathsf{Guess}]$ by the sum of these probabilities times the number of H queries.

On the other hand, in the simulation of Σ, δ_2 is given as the sum of $q_\Sigma \times \frac{1}{2^{k_1}}$ which represents the probability of the event that $(-1, *, *, *, *, *, r_{i_j}, *, *) \in$ $\mathsf{H\text{-}List}$ happens for random string r_{i_j} and we reject $\sigma_{i_{j-1}}$ whereas it is a valid signature (by accident), $q_\Sigma(\frac{q_G+q_H+q_\Sigma}{2^{k_1}})$ which represents the probability of the event that random string w_{i_j} is contained in $\mathsf{G\text{-}List}$, and $\frac{1}{2^N}$ which represents the failure in finding appropriate z_{i_j}.

The remaining term, $\beta^{q_\Sigma}\{\sum_{j=0}^{q_\Sigma}\Pr[\sharp L_i = j](1-\frac{1}{2^{k_0}})^j\}$ is maximized by choosing an optimal β in the same manner as for PSS (inequality (7) of [4]); we have

$$\beta^{q_\Sigma}\left\{\sum_{j=0}^{q_\Sigma}\Pr[\sharp L_i = j](1-\frac{1}{2^{k_0}})^j\right\} \geq \left(1+\frac{6q_\Sigma}{2^{k_0}}\right)^{-1}. \tag{2}$$

Therefore, Theorem 5 holds.

Efficient Secure Group Signatures with Dynamic Joins and Keeping Anonymity Against Group Managers

Aggelos Kiayias[1,⋆] and Moti Yung[2]

[1] Computer Science and Engineering,
University of Connecticut Storrs, CT, USA
aggelos@cse.uconn.edu

[2] RSA Laboratories, Bedford, MA, and Computer Science,
Columbia University New York, NY, USA
moti@cs.columbia.edu

Abstract. The demonstration of an efficient construction proven secure in a formal model that captures all intuitive security properties of a certain primitive is an ultimate goal in cryptographic design. This work offers the above for the case of a group signature scheme (with the traditional notion of dynamically joining users and untrusted join manager). To this end we adapt a formal model for group signatures capturing the state-of-the-art requirements in the area and we construct an efficient scheme and prove its security. Our construction is based on the scheme of Ateniese et al., which is modified appropriately so that it becomes provably secure. This task required designing novel cryptographic constructs as well as investigating some basic number-theoretic techniques for arguing security over the group of quadratic residues modulo a composite when its factorization is known. Along the way, we discover that in the basic construction, anonymity does not depend on factoring-based assumptions, which, in turn, allows the natural separation of user join management and anonymity revocation authorities. Anonymity can, in turn, be shown even against an adversary controlling the join manager.

1 Introduction

The notion of *group signature* is a useful anonymous non-repudiable credential primitive that was introduced by Chaum and Van Heyst [13]. This primitive involves a group of users, each holding a membership certificate that allows a user to issue a publicly verifiable signature which hides the identity of the signer within the group. The public-verification procedure employs only the public-key of the group. Furthermore, in a case of any dispute or abuse, it is possible for the group manager (GM) to "open" an individual signature and reveal the identity of its originator. Constructing an efficient and scalable group signature has been a research target for many years since its introduction with quite a slow

⋆ Research partly supported by NSF Career Award CNS-0447808.

progress, see e.g., [14,12,10,11,8,26,2,3,9,24,6]. The first construction in the literature that appeared to provide sufficient security as a general efficient scheme where user joins are performed by a manager that is not trusted to know their keys was the scalable scheme of Ateniese, Camenisch, Joye and Tsudik [3]. It provided constant signature size and resistance to attacks by coalitions of users. This scheme was based on a novel use of the DDH assumption combined with the Strong-RSA assumption over groups of intractable order. Recently, Bellare, Micciancio and Warinschi [4], noticing that [3] only prove a collection of individual intuitive security properties, advocated the need for a formal model for arguing the security of group signature. This basic observation is in line with the development of solid security notions in modern cryptography. They also offered a model of a relaxed group signature primitive and a generic construction in that model. Generic constructions are inefficient and many times are simpler than efficient constructions (that are based on specific number theoretic problems). This is due to the fact that generic constructions can employ (as a black box) the available heavy and powerful machinery of general zero-knowledge protocols and general secure multi-party computations. Thus, generic constructions typically serve only as plausibility results for the existence of a cryptographic primitive [20]. The relaxation in the model of [4] amounts to replacing dynamic adversarial join protocols where users get individual keys (as is the case in [3]) with a trusted party that generates and distributes keys securely.

The above state of affairs [3,4] indicates that there exists a gap in the long progression of research efforts regarding the group signature primitive. This gap is typical in cryptography and is formed by a difference between prohibitively expensive constructions secure in a formal sense and efficient more ad-hoc constructions. In many cases, as indicated above, it is easier to come up with provably secure generic inefficient constructions or to design efficient ad-hoc constructions. It is often much harder to construct an efficient implementation that is proven secure within a formal model (that convincingly captures all desired intuitive security properties). To summarize the above, it is apparent that the following question remained open:

> Design an **efficient** group signature with dynamic joins (and no trusted parties) which is **provably secure** within a formal model.

One of our contributions is solving the above open question by, both, adapting a new model for group signatures (based on the model of traceable signatures [23]), which follows the paradigm of [22] for the security of signature schemes, as well as providing an efficient provably secure construction (in the sense of the scheme of [3]).

This contribution reveals subtleties regarding what intractability assumptions are actually necessary for achieving the security properties. For example, the anonymity property in our treatment is totally disassociated from any factoring related assumption. (In order to reveal such issues a complete proof is needed following a concrete model, and this has not been done in the realm of (efficient) group signatures).

Our investigation also reveals delicate issues regarding the proper formal modeling of the group signature primitive with regards to the work of [4]. For example, the need of formalizing security against attacks by any internal or external entity that is active in the scheme (i.e., no trusted parties). Lack of such treatment, while proper for the relaxed notion of group signature of [4], is insufficient for proving the security of efficient state-of-the-art schemes that follow the line of work of [3].

Our Contributions. Below, we outline what this work achieves in more details.

1. Modeling. To model schemes like [3] with dynamic (yet sequential) joins and no trusted parties we adapt the model of [23] which is the first formal model in the area of group signing without added trusted parties. In particular, our model has the three types of attacks that involve the GM and the users as in [23]. We extend the model to allow adversarial opening of signatures (see the next item). All the attacks are modeled as games between the adversaries and a party called the interface. The interface represents the system in a real environment and simulates the behavior of the system (a probabilistic polynomial time simulator) in the security proof. The attacker gets oracle query capabilities to probe the state of the system and is also challenged with an attack task. We note that this follows the basic approach of [22] for modeling security of digital signatures and while not dealing with universal composability it nevertheless captures many natural protocol scenarios.

2. Adversarial Opening in Efficient Schemes. As mentioned above, our formal model extends the security requirements given by the list of security properties of [3] by allowing the adversary to request that the system opens signatures of its choice. In [3] opening of signatures was implicitly assumed to be an internal operation of the GM. We note that such stronger adversarial capability was put forth for the first time in the formal model of [4]. For achieving an efficient scheme with adversarial opening we needed to develop novel cryptographic constructs. (Note that adversarial opening can also be applied to strengthen the notion of traceable signatures).

3. Stronger Anonymity Property. In the scheme of [3] anonymity is argued against an adversary that is not allowed to corrupt the GM. This is a natural choice since in their scheme the GM holds the trapdoor which provides the opening capability, namely an ElGamal key. The GM also holds the trapdoor that is required to enroll users to the group, namely the factorization of an RSA-modulus. However, pragmatically, there is no need to combine the GM function that manages group members and allow them to join the group (which in real life can be run by e.g., a commercial company) with the opening authority function (which in real life can be run by a government entity). To manage members the GM who is the "Join Manager" still needs to know the factorization. The opening authority, on the other hand, must know the ElGamal key. This split of functions is not a relaxation of group signatures but rather a constraining of the primitive. In fact, now we should allow the anonymity adversary to *corrupt the GM as well.*

4. Number-Theoretic Results and Cryptographic Primitives. The last two contributions above required building cryptographic primitives over the set of quadratic residues modulo $n = pq$ that remain secure when the factorization (into two strong primes) p, q is known to the adversary.

To this end, we investigate the Decisional Diffie Hellman Assumption over the quadratic residues modulo n and we prove that it appears to be hard even if the adversary knows the factorization. In particular, we prove that any adversary that knows the factorization p, q and solves the DDH problem over the quadratic residues modulo a composite $n = pq$, can be turned into a DDH-distinguisher for quadratic-residues modulo a prime number. This result is of independent interest since it suggests that the DDH over $QR(n)$ does not depend to the factorization problem at all.

Also, the present work requires a CCA2 encryption mechanism that operates over the quadratic residues modulo n so that (i) encryption should not use the factorization of n, (i.e., the factorization need not be a part of the public-key), but on the other hand (ii) the factorization is *known* to the attacker. In this work we derive such a primitive in the form of an ElGamal variant following the general approach of twin encryption [28,16,19] which is CCA2 secure under the DDH assumption in the Random Oracle model (note that our efficient group signature requires the random oracle anyway since it is derived from the Fiat-Shamir transform [18,1]).

5. Efficient Construction. We provide an efficient construction of a group signature that is proven secure in our model. While, we would like to note that our scheme is strongly influenced by [3] (and originally we tried to rely on it as much as possible), our scheme, nevertheless, possesses certain subtle and important differences. These differences enable the proof of security of our scheme whereas the scheme in [3] cannot be proven secure in our model: There are many reasons for this, e.g., the scheme of [3] lacks an appropriate CCA2 secure identity embedding mechanism. Moreover, our efficient construction can support formally (if so desired), the separation of group management and opening capability something not apparent in the original scheme of [3]. Finally we note that a syntactically degenerated version of our construction (that retains its efficiency) can be proven secure in the model of [4] (and is, in fact, a relaxed group signature scheme of the type they have suggested).

An interesting result with respect to anonymity compared to previous work is highlighted in our investigation. Anonymity was argued in [3] to be based on the decisional Diffie-Hellman Assumption over Quadratic Residues modulo a composite and given that the GM was assumed to be uncorrupted, the key-issuing trapdoor (the factorization of the modulus) was not meant to be known to the adversary. As argued above, we prove that anonymity *still holds* when the adversary is given the factorization trapdoor. Thus, we disassociate anonymity from the factoring problem. Taking this result independently it also implies the separability between the opening authority and the group manager in the scheme of [3].

We note that the present work originally appeared as a technical report [25], where complete proofs can be found.

2 Preliminaries

Notations. We will write PPT for probabilistic polynomial-time. If $\mathcal{D}_1$ and $\mathcal{D}_2$ are two probability distributions defined over the same support that is parameterized by k we will write $\mathrm{dist}_{\mathcal{A}}(\mathcal{D}_1, \mathcal{D}_2)$ to denote the distance $|\mathbf{Prob}_{x \leftarrow \mathcal{D}_1}[\mathcal{A}(x) = 1] - \mathbf{Prob}_{x \leftarrow \mathcal{D}_2}[\mathcal{A}(x) = 1]|$. Note that typically $\mathrm{dist}_{\mathcal{A}}$ will be expressed as a function of k. If n is an integer, we will denote by $[n]$ the set $\{1, \ldots, n\}$. If we write $a \equiv_n b$ for two integers a, b we mean that n divides $a - b$ or equivalently that a, b are the same element within $\mathbf{Z}_n$. A function $f : \mathbb{N} \rightarrow \mathbb{R}$ will be called negligible if for all $c > 0$ there exists a k_c such that for all $k \geq k_c$, $f(k) < k^{-c}$. In this case we will write $f(k) = \mathsf{negl}(k)$. If $\ell, \mu \in \mathbb{N}$ we will write $S(2^\ell, 2^\mu)$ for the set $\{2^\ell - 2^\mu + 1, \ldots, 2^\ell + 2^\mu - 1\}$. PPT will stand for "probabilistic polynomial time." Throughout the paper (unless noted otherwise) we will work over the group of quadratic residues modulo n, denoted by $QR(n)$, where $n = pq$ and $p = 2p' + 1$ and $q = 2q' + 1$ and p, q, p', q' prime numbers. All operations are to be interpreted as modulo n (unless noted otherwise). In general we will use the letter ν to denote the security parameter (i.e., this value will be polynomially related to the sizes of all quantities involved). Next we define the Cryptographic Intractability Assumptions that will be relevant in proving the security properties of our constructions.

The first assumption is the Strong-RSA assumption. It is similar in nature to the assumption of the difficulty of finding e-th roots of arbitrary elements in $\mathbf{Z}_n^*$ with the difference that the exponent e is not fixed (i.e., it is not part of the instance).

Strong-RSA. Given a composite n (as described above), and $z \in QR(n)$, it is infeasible to find $u \in \mathbf{Z}_n^*$ and $e > 1$ such that $u^e = z (\bmod n)$, in time polynomial in ν.

Note that the variant we employ above restricts the input z to be a quadratic residue. This variant of Strong-RSA has been discussed before [15], and by restricting the exponent solutions to be only odd numbers we have that (i) it cannot be easier than the standard unrestricted Strong-RSA problem, but also (ii) it enjoys a random-self reducibility property (see [15]).

The second assumption that we employ is the Decisional Diffie-Hellman Assumption (see e.g., [5] for a survey). We state it below for a general group G and later on in definition 1 we will specialize this definition to two specific groups.

Decisional Diffie-Hellman. Given a description of a cyclic group G that includes a generator g, a DDH distinguisher $\mathcal{A}$ is a polynomial in ν time PPT that distinguishes the family of triples of the form $\langle g^x, g^y, g^z \rangle$ from the family of triples of the form $\langle g^x, g^y, g^{xy} \rangle$, where $x, y, z \in_R \#G$. The *DDH* assumption suggests that this advantage is a negligible function in ν.

Finally, we will employ the discrete-logarithm assumption over the quadratic residues modulo n with known factorization (note that the discrete-logarithm problem is assumed to be hard even when the factorization is known, assuming of course that the factors of n are large primes p, q and where $p-1$ and $q-1$ are non-smooth).

Discrete-Logarithm. Given two values a, b that belong to the set of quadratic residues modulo n with known factorization, so that $x \in [p'q'] : a^x = b$, find in time polynomial in ν the integer x so that $a^x = b$.

Conventions. (i) our proofs of knowledge will only be proven to work properly in the honest-verifier setting. On the one hand, the honest-verifier setting is sufficient for producing signatures. On the other hand, even in the general interactive setting the honest-verifier scenario can be enforced by assuming the existence, e.g., of a beacon, or some other mechanism that can produce trusted randomness; alternatively the participants may execute a coin flipping algorithm or use methods that transform the honest verifier proofs to a regular proofs. (ii) the public parameters employed in our various protocol designs (e.g., the composite modulus n) will be assumed to be selected honestly.

Discrete-log Relation Sets and Signatures. Discrete-log relation sets were introduced in [23] as a tool to describe proofs of zero-knowledge over the set of quadratic residues modulo n and will be useful in our designs. Informally a discrete-log relation set is specified by a $(m \times z)$-matrix that contains rows $\langle a_1^i, \ldots, a_m^i \rangle$ where each a_j^i is either an integer or a free variable drawn from a set of free variables $\{\alpha_1, \ldots, \alpha_r\}$. Each α_w has a range restriction imposed $\alpha_w \in S(2^{\ell_w}, 2^{\mu_w})$. A sequence of m elements $A_1, \ldots, A_m$ are part of the description as well. In [23] a proof of knowledge was described that allowed to an entity to prove knowledge of a set of r witnesses that, if substituted in all rows $\langle a_1^i, \ldots, a_m^i \rangle$ of the discrete-log relation matrix they would satisfy the relations $\prod_{j=1}^m A_i^{a_j^i} = 1$ as well as the range constraints. We refer to [23] for more details. In the 3-move protocol provided there the prover transmits z elements of $QR(n)$ denoted by $B_1, \ldots, B_z$ in the first move, receives a challenge $c \in \{0,1\}^k$ in the second move and transmits r integers $s_1, \ldots, s_r$ in the last round.

A digital signature based on a proof of knowledge based on a discrete-log relation set can be obtained by applying the Fiat-Shamir transform [18]. We recall briefly this transformation below. Let $\mathcal{G}$ be a hash function with range $\{0,1\}^k$ and D the matrix of some discrete-log relation set R over the base elements $A_1, \ldots, A_m \in QR(n)$. The proof of knowledge for discrete-log relation set can be made into a signature as follows: the signature on a message M will be denoted as $\mathtt{sgn}_{\mathcal{G}}^D(M)$ and computed as $\langle c, s_1, \ldots, s_r \rangle$ where $s_1, \ldots, s_r$ are computed as in the interactive proof using c equal to he hash $\mathcal{G}(M, A_1, \ldots, A_m, B_1, \ldots, B_z)$. The verification algorithm $\mathtt{ver}_{\mathcal{G}}^D$ on a signature $\langle c, s_1, \ldots, s_r \rangle$ for a message M is implemented by the following check: $c \overset{?}{=} \mathcal{G}(M, A_1, \ldots, A_m, B_1, \ldots, B_z)$, where each B_i is computed according to the verification routine of the interactive proof.

The security of the Fiat-Shamir signature construction [18] was investigated by [29] as was noted above.

Note that the proof of knowledge for discrete-log relation sets also enforces interval constraints on the witnesses. In particular when proving knowledge of a witness $x \in S(2^\ell, 2^\mu)$, the proof ensures that the witness belongs to the range $S(2^\ell, 2^{\epsilon(\mu+k)+2})$. This constraint comes "for free" in the soundness proof. If tighter integer ranges are needed they can also be achieved at the cost of making the proof slightly longer by employing the techniques of [7]. Nevertheless the tightness achieved by the proof for discrete-log relation sets itself will be sufficient for our designs.

3 DDH over $QR(n)$ with Known Factorization

Our constructions will require the investigation of the number-theoretic results presented in this section that albeit entirely elementary they have not being observed in the literature to the best of our knowledge. In particular we will show that DDH over $QR(n)$ does not depend on the hardness of factoring.

Let n be a composite, $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$ (p, q, p', q' primes). Recall that elements of $\mathbf{Z}_n^*$ are in a 1-1 correspondence with the set $\mathbf{Z}_p^* \times \mathbf{Z}_q^*$. Indeed, given $\langle b, c\rangle \in \mathbf{Z}_p^* \times \mathbf{Z}_q^*$, consider the system of equations $x \equiv b(\bmod p)$ and $x \equiv c(\mathrm{mod} q)$. Using Chinese remaindering we can construct a solution of the above system since $\gcd(p, q) = 1$ and the solution will be unique inside $\mathbf{Z}_n^*$. Alternatively for any $a \in \mathbf{Z}_n^*$ we can find the corresponding pair $\langle b, c\rangle$ in $\mathbf{Z}_p^* \times \mathbf{Z}_q^*$ by computing $b = a(\mathrm{mod} p)$ and $c = a(\mathrm{mod} q)$ (note that $\gcd(a, n) = 1$ implies that $b \not\equiv 0(\mathrm{mod} p)$ and $c \not\equiv 0(\mathrm{mod} q)$. The mapping ρ from $\mathbf{Z}_p^* \times \mathbf{Z}_q^*$ to $\mathbf{Z}_n^*$ is called the Chinese remaindering mapping. Observe that ρ preserves quadratic residuosity: $\rho(QR(p) \times QR(q)) = QR(n)$.

The following two lemmas will be useful in the sequel. They show (1) how the Chinese remaindering mapping behaves when given inputs expressed as powers inside the two groups $QR(p)$ and $QR(q)$, and (2) how discrete-logarithms over $QR(n)$ can be decomposed.

Lemma 1. *Let g_1, g_2 be generators of the groups $QR(p)$ and $QR(q)$ respectively, where the groups are defined as above. Then, if $\beta = \rho(g_1^{x_1}, g_2^{x_2})$, where ρ is the Chinese remaindering mapping, it holds that $\beta = \alpha^{q'x_1+p'x_2}(\mathrm{mod} n)$ where $\alpha = \rho(g_1^{(q')^{-1}}, g_2^{(p')^{-1}})$ is a generator of $QR(n)$.*

Lemma 2. *Fix a generator α of $QR(n)$ and an integer $t \in \mathbb{N}$. The mapping $\tau_\alpha : \mathbf{Z}_{p'} \times \mathbf{Z}_{q'} \to QR(n)$, with $\tau_\alpha(x_1, x_2) = \alpha^{(q')^t x_1+(p')^t x_2}$ is a bijection. The inverse mapping τ_α^{-1} is defined as $\tau_\alpha^{-1}(\alpha^x) = \langle (q')^{-t}x \mod p', (p')^{-t}x \mod q'\rangle$.*

Let $\mathrm{desc}(1^\nu)$ be a PPT algorithm, called a group descriptor, that on input 1^ν it outputs a description of a cyclic group G denoted by $\tilde{d}_G$. Depending on the group, $\tilde{d}_G$ may have many entries; in our setting it will include a generator of G, denoted by $\tilde{d}_G.\mathsf{gen}$ and the order of G denoted by $\tilde{d}_G.\mathsf{ord}$. We require that $2^{\nu-1} \leq \tilde{d}_G.\mathsf{ord} < 2^\nu$, i.e., the order of G is a ν-bit number with the first bit set. Additionally $\tilde{d}_G$ contains the necessary information that is required to implement multiplication over G. We will be interested in the following two group descriptors:

– $\mathrm{desc}_{\mathsf{p}}$: Given 1^ν find a ν-bit prime $p' > 2^{\nu-1}$ for which it holds that $p = 2p'+1$ and p is also prime. Let g be any non-trivial quadratic residue modulo p. We set $QR(p)$ to be the group of quadratic residues modulo p (which in this case is of order p' and is generated by g). The descriptor $\mathrm{desc}_{\mathsf{p}}$ returns $\langle g, p, p' \rangle$ and it holds that if $\tilde{d} \leftarrow \mathrm{desc}_{\mathsf{p}}(1^\nu)$, $\tilde{d}.\mathsf{ord} = p'$ and $\tilde{d}.\mathsf{gen} = g$.

– $\mathrm{desc}_{\mathsf{c}}$: Given ν find two distinct primes p', q' of bit-length $\nu/2$ so that $p'q'$ is a ν-bit number that is greater than $2^{\nu-1}$ and so that there exist primes p, q such that $p = 2p' + 1$ and $q = 2q' + 1$. Let g be any quadratic residue modulo n that is a generator of the group of $QR(n)$ (such element can be found easily). The descriptor $\mathrm{desc}_{\mathsf{c}}$ returns $\langle \alpha, n, p, q, p', q' \rangle$ and it holds that if $\tilde{d} \leftarrow \mathrm{desc}_{\mathsf{c}}(1^\nu)$, $\tilde{d}.\mathsf{ord} = p'q'$ and $\tilde{d}.\mathsf{gen} = \alpha$. The implementation of $\mathrm{desc}_{\mathsf{c}}$ that we will consider is the following: execute $\mathrm{desc}_{\mathsf{p}}$ twice, to obtain $\tilde{d}_1 = \langle g_1, p, p' \rangle$ and $\tilde{d}_2 = \langle g_2, q, q' \rangle$ with $p \neq q$, and set $\tilde{d} = \langle g, n = pq, p, q, p', q' \rangle$ where $\alpha = \rho(g_1^{(q')^{-1}}, g_2^{(p')^{-1}})$. For such a description $\tilde{d}$ we will call the descriptions $\tilde{d}_1$ and $\tilde{d}_2$, the prime coordinates of $\tilde{d}$.

Definition 1. *A Decisional Diffie Hellman (DDH) distinguisher for a group descriptor* desc *is a PPT algorithm $\mathcal{A}$ with range the set $\{0,1\}$; the advantage of the distinguisher is defined as follows:* $\mathsf{Adv}^{DDH}_{\mathrm{desc},\mathcal{A}}(\nu) = \mathrm{dist}_{\mathcal{A}}(\mathcal{D}^{\mathrm{desc}}_\nu, \mathcal{R}^{\mathrm{desc}}_\nu)$ *where* $\mathcal{D}^{\mathrm{desc}}_\nu$ *contains elements of the form* $\langle \tilde{d}, g^x, g^y, g^{x \cdot y} \rangle$ *where* $\tilde{d} \leftarrow \mathrm{desc}(1^\nu)$, $g = \tilde{d}.\mathsf{gen}$ *and* $x, y \leftarrow_R [\tilde{d}.\mathsf{ord}]$, *and* $\mathcal{R}^{\mathrm{desc}}_\nu$ *contains elements of the form* $\langle \tilde{d}, g^x, g^y, g^z \rangle$ *where* $\tilde{d} \leftarrow \mathrm{desc}(1^\nu)$, $g = \tilde{d}.\mathsf{gen}$ *and* $x, y, z \leftarrow_R [\tilde{d}.\mathsf{ord}]$. *Finally we define the overall advantage quantified over all distinguishers as follows:* $\mathsf{Adv}^{DDH}_{\mathrm{desc}}(\nu) = \max_{PPT\,\mathcal{A}} \mathsf{Adv}^{DDH}_{\mathrm{desc},\mathcal{A}}(\nu)$.

The main result of this section is the theorem below that shows that the DDH over $QR(n)$ *with known factorization* is essentially no easier than the DDH over the prime coordinates of $QR(n)$. The proof of the theorem is based on the construction of a mapping of DDH triples drawn from the two prime coordinate groups of $QR(n)$ into DDH triples of $QR(n)$ that is shown in the following lemma:

Lemma 3. *Let* $\tilde{d} \leftarrow \mathrm{desc}_{\mathsf{c}}(1^\nu)$ *with* $\tilde{d}_1, \tilde{d}_2 \leftarrow \mathrm{desc}_{\mathsf{p}}(1^{\nu/2})$, *its two prime coordinates, such that* $\tilde{d}_1 = \langle g_1, p, p' \rangle$ *and* $\tilde{d}_2 = \langle g_2, q, q' \rangle$. *The mapping ρ^* as follows:*

$$\rho^*(\langle \tilde{d}_1, A_1, B_1, C_1 \rangle, \langle \tilde{d}_2, A_2, B_2, C_2 \rangle) =_{\mathsf{df}} \langle \tilde{d}, \rho(A_1, A_2), \rho(B_1, B_2), \rho((C_1)^{q'}, (C_2)^{p'}) \rangle$$

satisfies the properties (i) $\rho^*(\mathcal{D}^{\mathrm{desc}_{\mathsf{p}}}_{\nu/2}, \mathcal{D}^{\mathrm{desc}_{\mathsf{p}}}_{\nu/2}) \cong \mathcal{D}^{\mathrm{desc}_{\mathsf{c}}}_\nu$ *and (ii)* $\rho^*(\mathcal{R}^{\mathrm{desc}_{\mathsf{p}}}_{\nu/2}, \mathcal{R}^{\mathrm{desc}_{\mathsf{p}}}_{\nu/2}) \cong \mathcal{R}^{\mathrm{desc}_{\mathsf{c}}}_\nu$, *where $\cong$ stands for statistically indistinguishable.*

The mapping ρ^* will return $\perp$ in case $\tilde{d}_1.\mathsf{ord} = \tilde{d}_2.\mathsf{ord}$. *This is a negligible probability event when selecting $\tilde{d}_1, \tilde{d}_2$ at random from* $\mathrm{desc}_{\mathsf{p}}$ *and is the event that contributes the negligible statistical difference in properties (i) and (ii).*

The lemma is used for the proof of the theorem below:

Theorem 1. $\mathsf{Adv}^{DDH}_{\mathrm{desc}_{\mathsf{c}}}(\nu) \leq 2\mathsf{Adv}^{DDH}_{\mathrm{desc}_{\mathsf{p}}}(\nu/2)$.

We proceed to state explicitly the two variants of the DDH assumption:

Definition 2. *The following are two Decisional Diffie Hellman Assumptions:*
• *The DDH assumption over quadratic residues for groups of prime order (*DDH-Prime*) asserts that:* $\mathsf{Adv}^{DDH}_{\mathrm{desc}_\mathsf{p}}(\nu) = \mathsf{negl}(\nu)$.
• *The DDH assumption over quadratic residues for groups of composite order with known Factorization (*DDH-Comp-KF*) asserts that:* $\mathsf{Adv}^{DDH}_{\mathrm{desc}_\mathsf{c}}(\nu) = \mathsf{negl}(\nu)$.

We conclude the section with the following theorem (where $\Longrightarrow$ stands for logical implication):

Theorem 2. DDH-Prime $\Longrightarrow$ DDH-Comp-KF.

4 CCA2 PK-Encryption over $QR(n)$ with Known Factorization

Our constructions will require an identity embedding mechanism that is CCA2 secure; such a mechanism is presented in this section.

A public-key encryption scheme comprises three procedures $\langle \mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec}\rangle$. The syntax of these procedures is as follows: $\mathtt{Gen}(1^\nu)$ returns a pair $\langle \mathsf{pk}, \mathsf{sk}\rangle$ that constitutes the public-key and secret-key of the scheme respectively. The probabilistic encryption function Enc takes as input the parameter 1^ν, a public-key pk and a message m and returns a ciphertext ψ. The decryption function Dec takes as input a secret-key sk and a ciphertext ψ and returns either the corresponding plaintext m, or the special failure symbol $\perp$. The soundness of a public-key encryption requires that for any $\langle \mathsf{pk}, \mathsf{sk}\rangle$, $\mathtt{Dec}(\mathsf{sk}, \mathtt{Enc}(1^\nu, \mathsf{pk}, m)) = m$ with very high probability in the security parameter ν (preferably always). There are various notions of security for public-key encryption [21,28,30,17], below we will be interested in the so-called CPA and CCA2 security in the indistinguishability sense.

Now consider the following cryptosystem $\langle \mathtt{Gen}_{qr}, \mathtt{Enc}_{qr}, \mathtt{Dec}_{qr}\rangle$:

– The key-generator $\mathtt{Gen}_{qr}$ on input 1^ν samples the description $\tilde{d} = \langle g, n, p, q, p', q'\rangle \leftarrow \mathrm{desc}_\mathsf{c}(1^\nu)$, selects a value $x \leftarrow_R [p'q']$ and outputs $\mathsf{pk} = \langle g, n, p, q, h = g^x\rangle$ and $\mathsf{sk} = x$.
– The encryption function $\mathtt{Enc}_{qr}$ operates as follows: given $M \in QR(n)$, it selects $r \leftarrow_R [\lfloor n/4 \rfloor]$ and returns the pair $\langle g^r, h^r M\rangle$.
– The decryption operation $\mathtt{Dec}_{qr}$ is given $\langle G, H\rangle$ and returns $G^{-x}H(\bmod n)$.

Note that this cryptosystem is an ElGamal variant over quadratic residues modulo a composite, so that (i) the factorization is available to the adversary, but: (ii) the factorization is not necessary for encryption.

Theorem 3. *The cryptosystem* $\langle \mathtt{Gen}_{qr}, \mathtt{Enc}_{qr}, \mathtt{Dec}_{qr}\rangle$ *described above satisfies* CPA*-security under the assumption* DDH-Compo-KF*, and thus under the assumption* DDH-Prime *(theorem 2).*

We remark that ElGamal variants over composite order groups have been considered before, e.g., [27]; in the setup that was considered the adversary was denied the factorization and security properties of the cryptosystem were associated with the factoring assumption. Our variant above, on the other hand, shows that the semantic security (in the sense of CPA-security) of the composite modulus ElGamal variant we define still holds under the standard prime-order Decisional Diffie-Hellman assumption DDH-Prime.

Now let us turn our attention to achieving CCA2 security in the above setting. To achieve this goal we will employ double encryption. Double encryption has been employed as a tool to obtain chosen-ciphertext security originally in [28]. The so called "twin-conversion" has been formalized in [19] and transforms a CPA-secure cryptosystem into a CCA2-cryptosystem This result applies directly to our cryptosystem $\langle \mathtt{Gen}_{qr}, \mathtt{Enc}_{qr}, \mathtt{Dec}_{qr} \rangle$ as defined above. We omit the details for the full version.

5 Group Signatures: Model and Definitions

The parties that are involved in a group signature scheme are the Group Manager (GM) and the users. In the definition below we give a formal syntax of the five procedures the primitive is based on.

Our formalization is geared towards schemes as the scheme of [3] where users are joining the system by executing a join-dialog with the GM (and not any other trusted entity or tamper-proof element exists). Naturally, this formalization can capture *also* the case where a third party creates the user signing keys privately and distributes them through private channels and with trusted parties, however we do not deal with this easier case in our model.

Definition 3. *A group signature scheme is a digital signature scheme that comprises of the following five procedures;*

SETUP*: On input a security parameter 1^ν, this probabilistic algorithm outputs the group public key $\mathcal{Y}$ (including all system parameters) and the secret key $\mathcal{S}$ for the GM. Note that* SETUP *is* not *supposed to output the members' signing keys. Moreover* SETUP *initializes a public-state string St with two components $St_{users} = \emptyset$ (a set data structure) and $St_{trans} = \epsilon$ (a string data structure).*

JOIN*: A* protocol *between the GM and a user that results in the user becoming a new group member. The user's output is a membership certificate and a membership secret. We denote the i-th user's membership certificate by* cert_i *and the corresponding membership secret by* sec_i*. Since* JOIN *is a protocol, it is made out of two interactive Turing Machines (ITM)* $\mathsf{J}_{\mathsf{user}}, \mathsf{J}_{\mathsf{GM}}$*. Only* $\mathsf{J}_{\mathsf{user}}$ *has a private output tape. An execution of the protocol is denoted as* $[\mathsf{J}_{\mathsf{user}}(1^\nu, \mathcal{Y}), \mathsf{J}_{\mathsf{GM}}(1^\nu, St, \mathcal{Y}, \mathcal{S})]$ *and has two "output" components: the private output of the user,* $\langle i, \mathsf{cert}_i, \mathsf{sec}_i \rangle \leftarrow \mathsf{U}[\mathsf{J}_{\mathsf{user}}(1^\nu, \mathcal{Y}), \mathsf{J}_{\mathsf{GM}}(1^\nu, St, \mathcal{Y}, \mathcal{S})]$ *and the public transcript,* $\langle i, \mathsf{transcript}_i \rangle \leftarrow \mathsf{T}[\mathsf{J}_{\mathsf{user}}(1^\nu, \mathcal{Y}), \mathsf{J}_{\mathsf{GM}}(1^\nu, St, \mathcal{Y}, \mathcal{S})]$*. After a successful execution of* JOIN *the following (public) updates are made to the state: $St_{users} = St_{users} \cup \{i\}$ and $St_{trans} = St_{trans} ||$* $\langle i, \mathsf{transcript}_i \rangle$*. (Note: the identity of the user that gets the i-th user's membership certificate is assumed to be authenticated and thus associated with i; the GM invokes the JOIN procedure one user at a time).*

$\texttt{SIGN}$*: A probabilistic algorithm that given a group's public-key, a membership certificate, a membership secret, and a message m outputs a signature for the message m. We write* $\texttt{SIGN}(\mathcal{Y}, \mathsf{cert}_i, \mathsf{sec}_i, m)$ *to denote the application of the signing algorithm.*

$\texttt{VERIFY}$*: An algorithm for establishing the validity of an alleged group signature of a message with respect to a group public-key. If σ is a signature on a message m, then we have* $\texttt{VERIFY}(\mathcal{Y}, m, \sigma) \in \{\top, \bot\}$.

$\texttt{OPEN}$*: An algorithm that, given a message, a valid group signature on it, a group public-key, the GM's secret-key and the public-state it determines the identity of the signer. In particular* $\texttt{OPEN}(m, \sigma, \mathcal{Y}, \mathcal{S}, St) \in St_{users} \cup \{\bot\}$.

Notation. We will write $\langle i, \mathsf{cert}_i, \mathsf{sec}_i \rangle \leftrightharpoons_{\mathcal{Y}} \langle i, \mathsf{transcript}_i \rangle$ to denote the relationship between the private output of J_{user} and the public-transcript when the protocol is executed based on the group public-key $\mathcal{Y}$ and a state St (note that we omit St in the subscript for convenience). Moreover, any given cert, based on a public-key $\mathcal{Y}$, has a corresponding sec; we will also denote this relationship by $\mathsf{cert} \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}$ (overloading the notation). We remark that $\leftrightharpoons_{\mathcal{Y}}$ in both cases, will be considered a polynomial-time relationship in the parameter ν.

Given a $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \texttt{SETUP}(1^{\nu})$, a public-state St is called *well-formed* if it is effectively produced by a Turing machine M that has unlimited access to a J_{GM} oracle (following the public state update procedures as in definition 3). A well-formed state St' is said to extend state St, if it is effectively produced by a Turing machine as above but with the public-state initially set to St instead of $\langle \emptyset, \epsilon \rangle$.

Correctness. The correctness of a group signature scheme is broken down in four individual properties: (i) *user tagging soundness* mandates that users are assigned a unique tag (depending on order of joining) by the $\texttt{JOIN}$ protocol; (ii) *join soundness* mandates that the private output tape of $\mathsf{J}_{\mathsf{user}}$ after a successful execution of the $\texttt{JOIN}$ dialog contains a valid membership certificate and membership secret; (iii) *signing soundness* mandates that the group signature scheme behaves like a digital signature; (iv) *opening soundness* mandates that the $\texttt{OPEN}$ algorithm succeeds in identifying the originator of any signature generated according to specifications. Formally,

Definition 4. *A group signature is correct if the following statements hold with very high probability over the coin tosses of all procedures. Let* $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \texttt{SETUP}(1^{\nu})$.

- User tagging soundness. *In every well formed public-state St it holds that the cardinality of the set St_{users} equals the number of transcripts in the string St_{trans}.*
- Join soundness. *If* $\langle i, \mathsf{cert}_i, \mathsf{sec}_i \rangle \leftarrow \mathsf{U}[\ \mathsf{J}_{\mathsf{user}}(1^{\nu}, \mathcal{Y}), \mathsf{J}_{\mathsf{GM}}(1^{\nu}, St, \mathcal{Y}, \mathcal{S})]$ *then it holds that* $\mathsf{cert}_i \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}_i$.
- Signing soundness. *For any* $\mathsf{cert} \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}$*, and any message m,* $\texttt{VERIFY}(\mathcal{Y}, m, \texttt{SIGN}(\mathcal{Y}, \mathsf{cert}, \mathsf{sec}, m)) = \top$.
- Opening soundness. *For any certificates, transcripts and well-formed public-state St s.t.* $\langle i, \mathsf{cert}_i, \mathsf{sec}_i \rangle \leftrightharpoons_{\mathcal{Y}} \langle i, \mathsf{transcript}_i \rangle$*, if St' is a well-formed public-state that extends St with* $\langle i, \mathsf{transcript}_i \rangle \in St'_{trans}$*, then for any message m, and any* $\sigma \leftarrow \texttt{SIGN}(\mathcal{Y}, \mathsf{cert}_i, \mathsf{sec}_i, m)$ *it holds that* $\texttt{OPEN}(m, \sigma, \mathcal{Y}, \mathcal{S}, St') = i$.

Security. Below we present the general model for security. A number of oracles are specified. Through these oracles the adversary may interact with an Interface that represents the system in the real world, and simulates its operation (i.e., a simulator) in the security proof. This allows us to model adversaries with capabilities (modeled by subsets of the oracles) and attack goals in mind, in the spirit of [22]. However, since we deal with a "privacy primitive" we have to deal with a number of goals of mutually distrusting and mutually attacking parties, thus we need more than one adversarial scenario. The interface $\mathcal{I}$ is an ITM that employs a data structure called state $\mathsf{state}_{\mathcal{I}}$ and is initialized as $\langle St, \mathcal{Y}, \mathcal{S}\rangle \leftarrow \mathtt{SETUP}(1^{\nu})$. The interface accepts the types of queries listed below. We remark that during an attack the adversary interacts with the interface and the oracles in a stateful fashion and the interface performs a number of bookkeeping operations that involve $\mathsf{state}_{\mathcal{I}}$ as explained below.

- $\mathcal{Q}_{\mathsf{pub}}$ and $\mathcal{Q}_{\mathsf{key}}$: the interface looks up $\mathsf{state}_{\mathcal{I}}$ and returns the public-and secret-key respectively.
- $\mathcal{Q}_{\mathsf{a-join}}$: the interface initiates a protocol dialog simulating J_{GM}. The user created from this interaction (if it is successfuly terminated) will be entered in St_{users} and the transcript in St_{trans} following the updating rules of definition 3. Additionally the user will be marked as U^a (adversarially controlled).
- $\mathcal{Q}_{\mathsf{b-join}}$: the interface initiates a protocol dialog simulating $\mathsf{J}_{\mathsf{user}}$. The user created from this interaction (if successfully terminated) will be entered in St_{users} and the transcript into St_{trans} as described in the update procedure of definition 3. Additionally, the user will be marked by U^b. Upon successful termination the resulting membership certificate and membership secret (i.e., the output of the user) will be appended in a **private** area of $\mathsf{state}_{\mathcal{I}}$.
  Following the above we note that the adversary when executing the $\mathcal{Q}_{\mathsf{b-join}}$ query will be effectively required by the interface to choose a unique and properly defined tag for the current user (according to definition 3). This is not a restriction since this can be enforced in practice by having the user checking the public user name database during normal protocol executions [also note: it is assumed here that the user name database St_{users} is not entirely adversarially controlled; indeed, if St_{users} is compromised then clearly no group signature scheme can have any form of identification robustness].
- $\mathcal{Q}_{\mathsf{read}}, \mathcal{Q}_{\mathsf{write}}$: these two queries allow to the adversary to read and write respectively $\mathsf{state}_{\mathcal{I}}$. The query $\mathcal{Q}_{\mathsf{read}}$ returns the whole $\mathsf{state}_{\mathcal{I}}$ excluding the public and secret-key as well as the private area of $\mathsf{state}_{\mathcal{I}}$ that is used for the $\mathcal{Q}_{\mathsf{b-join}}$ queries. The query $\mathcal{Q}_{\mathsf{write}}$ is allowed to perform arbitrary changes as long as it does not remove/corrupt elements from St_{users}, St_{trans} (but e.g., insertion to these structures is allowed).
- $\mathcal{Q}_{\mathsf{sign}}(i, m)$: given that $i \in U^b$ the interface simulates a signature on m by looking up the membership certificate and membership secret available in the private area of $\mathsf{state}_{\mathcal{I}}$ and returns a corresponding signature.
- $\mathcal{Q}_{\mathsf{open}}(\sigma)$: the interface applies the opening algorithm to the given signature σ using the current St. If S is a set of signatures we denote by $\mathcal{Q}_{\mathsf{open}}^{\neg S}$ the operation of the opening oracle when queries for signatures in S are declined.

We remark that the interface $\mathcal{I}$ maintains a history of all queries posed to the above oracles (if these queries accepted an input); for instance, we use the notation $\mathsf{hist}_{\mathcal{I}}(\mathcal{Q}_{\mathsf{sign}})$ to denote the history of all signature queries.

Security Modeling. We next define our security model, which involves three attack scenarios and corresponding security definitions. These security properties are based on our modeling of Traceable Signatures [23] and are ported from the traceable signature setting to the group signature setting, augmenting them with adversarial opening capability. In particular, we use the same terminology for the attacks to facilitate the comparison between these two primitives.

The first security property relates to an adversary that wishes to misidentify itself. In a misidentification-attack the adversary is allowed to join the system through $\mathcal{Q}_{\mathsf{a-join}}$ queries and open signatures at will; finally he produces a forged group signature (cf. an existential adaptive chosen message attack, [22]) that does not open into one of the users he controls (actually without loss of generality the adversary controls all users of the system; thus the adversary wins if the opening algorithm returns $\perp$).

```
The Misidentification-Attack Game $G^{\mathcal{A}}_{\mathsf{mis}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{mis}}(1^{\nu})$):
1. $\mathsf{state}_{\mathcal{I}} = \langle St, \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^{\nu})$;
2. $\langle m, \sigma \rangle \leftarrow \mathcal{A}^{\mathcal{I}[\mathcal{Q}_{\mathsf{pub}}, \mathcal{Q}_{\mathsf{a-join}}, \mathcal{Q}_{\mathsf{read}}, \mathcal{Q}_{\mathsf{open}}]}(1^{\nu})$
3. $i = \mathtt{OPEN}(m, \sigma, \mathcal{Y}, \mathcal{S}, St)$
4. If $(\mathtt{VERIFY}(\mathcal{Y}, m, \sigma) = \top) \wedge (i \notin U^a)$ then return $\top$ else return $\perp$.
```

Our second security property relates to a framing type of attack. Here the whole system conspires against the user. The adversary is in control not only of coalitions of users but of the GM itself. It is allowed to introduce "good" users into the system by issuing $\mathcal{Q}_{\mathsf{b-join}}$ queries to the interface and obtain signatures from them. Finally the adversary produces a signature that opens to one of the "good" users. Note that the adversary can take advantage of $\mathcal{Q}_{\mathsf{write}}$ to create dummy users if it so wishes.

```
The Framing-Attack Game $G^{\mathcal{A}}_{\mathsf{fra}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{fra}}(1^{\nu})$):
1. $\mathsf{state}_{\mathcal{I}} = \langle St, \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^{\nu})$;
2. $\langle m, \sigma \rangle \leftarrow \mathcal{A}^{\mathcal{I}[\mathcal{Q}_{\mathsf{pub}}, \mathcal{Q}_{\mathsf{key}}, \mathcal{Q}_{\mathsf{b-join}}, \mathcal{Q}_{\mathsf{read}}, \mathcal{Q}_{\mathsf{write}}, \mathcal{Q}_{\mathsf{sign}}]}(1^{\nu})$
3. $i = \mathtt{OPEN}(m, \sigma, \mathcal{Y}, \mathcal{S}, St)$
4. If $(\mathtt{VERIFY}(\mathcal{Y}, m, \sigma) = \top) \wedge (i \in U^b) \wedge ((i, m) \notin \mathsf{hist}_{\mathcal{I}}(\mathcal{Q}_{\mathsf{sign}}))$ then return
   $\top$ else return $\perp$.
```

Finally we model anonymity. In an anonymity-attack the adversary operates in two stages play and guess. In the play stage the adversary is allowed to join the system through $\mathcal{Q}_{\mathsf{a-join}}$ queries, as well open signatures through $\mathcal{Q}_{\mathsf{open}}$ queries. The adversary terminates the play stage by providing a pair of membership certificates/secrets (that were possibly obtained through $\mathcal{Q}_{\mathsf{a-join}}$ queries). The adversary obtains a "challenge signature" using one of the two membership certificate/secrets it provided at random, and then proceeds in the guess stage that operates identically to the play stage with the exception that the adversary is not allowed to open the challenge signature. Note that this attack is similar to a CCA2 attack when an individual group signature is considered an identity concealing ciphertext.

```
The Anonymity-attack Game $G^{\mathcal{A}}_{\mathsf{anon}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{anon}}(1^{\nu})$):
1. $\mathsf{state}_{\mathcal{I}} = \langle St, \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^{\nu})$;
2. $\langle aux, m, \mathsf{cert}_1, \mathsf{sec}_1, \mathsf{cert}_2, \mathsf{sec}_2, \rangle \leftarrow \mathcal{A}^{\mathcal{I}[\mathcal{Q}_{\mathsf{pub}}, \mathcal{Q}_{\mathsf{a-join}}, \mathcal{Q}_{\mathsf{read}}, \mathcal{Q}_{\mathsf{open}}]}(\mathsf{play}, 1^{\nu})$
3. if $\neg((\mathsf{cert}_1 \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}_1) \wedge (\mathsf{cert}_2 \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}_2))$ then terminate and return $\perp$;
4. Choose $b \leftarrow_R \{1, 2\}$;
5. $\sigma \leftarrow \mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}_b, \mathsf{sec}_b, m)$;
6. $b^* \leftarrow \mathcal{A}^{\mathcal{I}[\mathcal{Q}_{\mathsf{pub}}, \mathcal{Q}_{\mathsf{a-join}}, \mathcal{Q}_{\mathsf{read}}, \mathcal{Q}^{\neg\{\sigma\}}_{\mathsf{open}}]}(\mathsf{guess}, aux)$;
7. if $b = b^*$ return $\top$ else return $\perp$;
```

Definition 5. *A group signature scheme is secure if for all PPT $\mathcal{A}$ it holds that (i)* $\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{mis}}(1^{\nu}) = \top] = \mathsf{negl}(\nu)$ *(ii)* $\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{fra}}(1^{\nu}) = \top] = \mathsf{negl}(\nu)$ *and (iii)* $2\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{anon}}(1^{\nu}) = \top] - 1 = \mathsf{negl}(\nu)$.

Capturing the intuitive security properties of [3]. Given the above security model is relatively straightforward to see that the informal security properties that were put forth in [3] are captured by the above three security properties. In particular, (1) *Unforgeability*: an adversary that given the public-key forges a signature, will either produce a signature that opens to $\perp$ or a signature that opens to one of the users; such an attack is prevented by both misidentification and framing security above; (2) *Anonymity*, is captured by the anonymity security property above, (3) *Unlinkability*, is also captured by the anonymity security property, (4) *Exculpability* is captured by framing security (since the secret-key of the GM is released to the adversary), (5) *Traceability*, is ensured by misidentification (a signer cannot produce a signature that opens to $\perp$) and framing security (a signer cannot frame another user). (6) *Coalition resistance* is built-in into our security properties since w.r.t. misidentification we allow the adversary to adaptively build a coalition of malicious users, whereas in the case of framing attack the adversary has the GM's key (and as a result it can build a coalition if it wishes it).

6 Building a Secure Group Signature

The scheme we will present will be built based on the state-of-the-art scheme of [3]. We note again that it is impossible to prove security of the scheme of [3] in our model (yet we insist on minimally changing it to achieve the provable properties).

The public-parameters of the group signature are a composite modulus n of ν bits, such that $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$ (where p, q, p', q' are primes), as well as a sequence of elements inside $QR(n)$ denoted by a_0, a, g, y and two lengths ℓ, μ, so that $S(2^{\ell}, 2^{\mu}) \subseteq \{1, \ldots, p'q'\}$. The membership certificates are of the form $\langle A, e \rangle$ so that $A \in QR(n)$ and e is a prime number in $S(2^{\ell}, 2^{\mu})$. The membership secret is a value x such that $a_0 a^x = A^e$. Given the above structure, the basic functions of the group signature scheme employ two hash functions $\mathcal{G}, \mathcal{H}$ and are implemented as follows:

SETUP: On input a security parameter ν, this probabilistic algorithm first samples a group description for $\langle g, n, p, q, p', q' \rangle \leftarrow \mathrm{desc}_{\mathsf{c}}(1^\nu)$. Then, it selects $x, \hat{x} \leftarrow_R \mathbf{Z}^*_{p'q'}$, $a_0, a, h \leftarrow_R QR(n)$ and publishes the group public key $\mathcal{Y} =_{\mathsf{df}} \langle n, a_0, a, g, h, y = g^x, \hat{y} = g^{\hat{x}} \rangle$ and the secret key is set to $\mathcal{S} =_{\mathsf{df}} \langle p, q, x, \hat{x} \rangle$. The procedure also selects the parameters $\ell, \mu, k \in \mathbb{N}$ and $\epsilon > 1$ as functions of ν so that this condition is satisfied $S(2^\ell, 2^{\epsilon(\mu+k)+2}) \subseteq \{5, \ldots, \min\{p', q'\} - 1\}$.

JOIN: A protocol between the GM and a user that allows the joint computation of a membership certificate $\langle A, e \rangle$ so that only the user obtains the membership secret x. First we give the functionality of the protocol using a trusted party T: the specification of the protocol $\mathsf{J}^T_{\mathsf{user}}, \mathsf{J}^T_{\mathsf{GM}}$ using a third trusted party T is as follows: $\mathsf{J}^T_{\mathsf{user}}(1^\nu, \mathcal{Y})$ sends "go" to the trusted party T, who in turn selects $x \leftarrow_R \lfloor n/4 \rfloor$ and writes to the GM's communication tape the value $C = a^x \bmod n$ and writes to the user's private tape the value x. $\mathsf{J}^T_{\mathsf{GM}}(1^\nu, \mathcal{Y}, \mathcal{S})$ reads C from the communication tape with T, it selects a prime $e \leftarrow_R S(2^\ell, 2^\mu) - \{p', q'\}$ and computes $A = (a_0 a)^{1/e} (\mathrm{mod} n)$; finally it writes $\langle i, A, e \rangle$ in the regular communication tape where i is the next available user tag (a counter is employed) and terminates. $\mathsf{J}^T_{\mathsf{user}}$ reads $\langle A, e \rangle$ from the communication tape and writes $\langle i, A, e, x \rangle$ in its private output tape. As shown in the "non-adaptive drawings of random powers" protocol of (section 6, [23]) it is possible to derive an *efficient* protocol $\mathsf{J}_{\mathsf{user}}, \mathsf{J}_{\mathsf{GM}}$ that *does not* employ a trusted party and achieves the above ideal functionality. We remark that the GM is accepting join protocols only in a sequential fashion.

In the above description, $\mathsf{cert}_i = \langle A, e \rangle$, $\mathsf{sec}_i = x$, $\mathsf{transcript}_i = \langle i, C, A, e \rangle$. If $\mathsf{transcript} = \langle i_t, C_t, A_t, e_t \rangle$ and $\mathsf{cert} = \langle A_c, e_c \rangle$, $\mathsf{sec} = x_c$, the relationship $\mathsf{cert} \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}$ is true iff $A_c^{e_c} = a_0 a^{x_c} (\bmod n)$, and the relationship $\langle i, \mathsf{transcript} \rangle \leftrightharpoons_{\mathcal{Y}} \langle i, \mathsf{cert}, \mathsf{sec} \rangle$ is true iff $i_t = i$, $A_t = A_c$, $e_t = e_c$ and $\mathsf{cert} \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}$.

SIGN: The signing algorithm is based on a proof of knowledge that is preceded by the values $\langle T_1, T_2, \hat{T}_1, \hat{T}_2, T_3, T_4 \rangle$ defined as follows when invoked by the i-th user:

$$r, \hat{r} \leftarrow_R \lfloor n/4 \rfloor \ : \ T_1 = A_i y^r, \ T_2 = g^r, \ \hat{T}_1 = A_i \hat{y}^{\hat{r}}, \ \hat{T}_2 = g^{\hat{r}}, \ T_3 = g^{e_i} h^r$$

$$T_4 = \mathtt{nizk}^{\mathcal{H}}[n, g, y_1, y_2, \langle T_2, T_1 \rangle, \langle \hat{T}_2, \hat{T}_1 \rangle]$$

The noninteractive proof of knowledge T_4 ensures that the twin ciphertext $T_1, T_2, \hat{T}_1, \hat{T}_2$ is properly formed (i.e., that the plaintext is the same). To complete the description of the signature, we design a discrete-log relation set over the free variables r, e, x, s', s'', to prove the following relations: $T_2 = g^r, T_3 = g^e h^r, T_2^e = g^{s'}, a_0 a^x y^{s'} = T_1^e, T_3 = g(g^2)^{s''} h^r$. This proof ensures that T_1, T_2 is the ElGamal encryption of a value A that if raised to an odd integer e, it can be split by the prover in the form $a_0 a^x$. The signature on a message M will be formed by employing the Fiat-Shamir transform over the proof of knowledge in the discrete-log relation set as described in section 2. We defer more details for the full version, we only remark that the proof will be a tuple of the form $\langle c, s_1, s_2, s_3, s_4, s_5 \rangle$ (in addition to the elements $T_1, T_2, \hat{T}_1, \hat{T}_2, T_3, T_4$).

VERIFY: given a signature $\sigma = \langle T_1, T_2, \hat{T}_1, \hat{T}_2, T_3, T_4, c, s_1, s_2, s_3, s_4, s_5 \rangle$ the verification algorithm will verify the proof of knowledge for the discrete-log relation set as well as verify T_4.

OPEN: The opening procedure given a signature σ is as follows:

1. Verify σ using the public verification procedure VERIFY.
2. Parse σ to recover the values T_1, T_2.
3. Verify that the noninteractive proof of knowledge T_4 is correct.
4. Compute $A = T_1(T_2^x)^{-1} \bmod n$.
5. Match A to some user's first component of the membership certificate $\langle A_i, e_i \rangle$ (as available in the database maintained during the JOIN protocols).
6. If either steps 1 or 3 or 5 fail, return $\perp$, else return the user found in step 5.

Theorem 4. *The group signature* $\langle$SETUP, JOIN, SIGN, VERIFY, OPEN$\rangle$ *defined above is (i) correct,(ii) satisfies security against misidentification attacks under the Strong-RSA assumption in the random oracle model, (iii) satisfies security against framing attacks under the Discrete-logarithm assumption over the $QR(n)$ with known factorization, in the random oracle model, and (iv) satisfies security against anonymity-attacks, under the* DDH-Compo-KF *in the random oracle model.*

7 Group Signatures with Authority Separability: Anonymity Against the GM

In a group signature with separated authorities we differentiate between the GM, who is responsible for group membership operations and an Opening Authority (OA), who is responsible for the revocation of anonymity (opening a signature). This separation is relevant to practice, since group management should be typically considered an ISP operation whereas revocation of anonymity must be performed by some (possible external) third-party authority (which can even be distributed). This authority separability is natural and is not designed to assure that certain processes are tamper-proof; note that it is a different (weaker) notion of separability compared to what [11] considered (who considered the full disassociation of all involved parties). The extension of the present formal model to stronger notions of separability [26] is possible. Nevertheless in this case we are interested in what can be achieved without incurring *any* additional cost at our basic construction. Stronger notions of separability can be achieved nevertheless at additional costs (both in terms of communication and computation).

The syntax of a group signature with authority separability is similar to the group signature syntax as presented in definition 3 with the modifications:

Definition 6. *A group signature scheme with authority separability is a digital signature scheme comprising the following six procedures; the parties involved are the GM, the opening authority and the users.*

SETUP$_{\mathsf{GM}}$*: On input a security parameter 1^ν, this probabilistic algorithm outputs the group public key $\mathcal{Y}_{\mathsf{GM}}$ (including necessary system parameters) and the* *secret key $\mathcal{S}_{\mathsf{GM}}$ for the GM.* `SETUP`$_{\mathsf{GM}}$ *also initializes a public-state string St with two components $St_{users} = \emptyset$ and $St_{trans} = \epsilon$.*

`SETUP`$_{\mathsf{OA}}$*: On input a security parameter 1^ν, and the public-key $\mathcal{Y}_{\mathsf{GM}}$, this probabilistic algorithm generates the public and secret-key of the opening authority denoted by $\mathcal{Y}_{\mathsf{OA}}$ and $\mathcal{S}_{\mathsf{OA}}$.*

We will denote the concatenation of $\mathcal{Y}_{\mathsf{OA}}$ and $\mathcal{Y}_{\mathsf{GM}}$ by $\mathcal{Y}$.

`JOIN`*: The* `JOIN` *protocol is identical to that of definition 3 with the only exception J_{GM} requires only the secret key of the GM, $\mathcal{S}_{\mathsf{GM}}$.*

`SIGN`*: identical to definition 3.*

`VERIFY`*: identical to definition 3.*

`OPEN`*: the opening algorithm is the same as in definition 3 with the exception that only the opening authority's secret-key $\mathcal{S}_{\mathsf{OA}}$ is required.*

Note that above we consider that the setup procedure for the OA acts on the public-key of the GM. While our construction below will take advantage of this syntactic condition, it is not hard in general to avoid it at the expense of extending the length of the signature by a constant amount (and thus separate the GM and OA even in the setup phase).

Correctness. Given the above minor syntactic differences, the correctness of a group-signature with separated authorities is defined in the same way as definition 4 by taking into account the above modifications that correspond to the fact that J_{GM} requires only $\mathcal{S}_{\mathsf{GM}}$ and `OPEN` requires only $\mathcal{S}_{\mathsf{OA}}$.

Security. The security properties of a group-signature with separated authorities must remain the same so that any secure group signature with separated authorities must also be a secure group signature (by collapsing the GM and the OA into a single entity). Moreover in the separated authority setting the anonymity-attack can be made even stronger by *adding* the adversarial capability of corrupting the GM.

Regarding the security modeling, in the queries that can be posed to the interface, the query $\mathcal{Q}_{\mathsf{key}}$ will be substituted with two distinct queries $\mathcal{Q}_{\mathsf{keyGM}}$ and $\mathcal{Q}_{\mathsf{keyOA}}$ with the obvious results. The definition of the three attacks will remain unaltered with the following syntactic modifications:

(i) in a framing-attack the adversary will have at its disposal both the queries $\mathcal{Q}_{\mathsf{keyGM}}$ and $\mathcal{Q}_{\mathsf{keyOA}}$ (i.e., the adversary can corrupt *both* the GM and the OA)
(ii) in the anonymity attack, the adversary will be given *additional* access to the $\mathcal{Q}_{\mathsf{keyGM}}, \mathcal{Q}_{\mathsf{write}}$ queries in both phases of the attack game.

The above two modifications are straightforward and thus we will not list the security properties again in this section. The modified games will be denoted by $G^{\mathcal{A}}_{\mathsf{fra-sep}}, G^{\mathcal{A}}_{\mathsf{mis-sep}}, G^{\mathcal{A}}_{\mathsf{anon-sep}}$.

Definition 7. *A group signature scheme with separated authorities is secure if for all PPT $\mathcal{A}$ it holds that (i)* $\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{in-sep}}(1^\nu) = \top] = \mathsf{negl}(\nu)$ *as well as (ii)* $\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{out-sep}}(1^\nu) = \top] = \mathsf{negl}(\nu)$ *and (iii)* $2\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{anon-sep}}(1^\nu) = \top] - 1 = \mathsf{negl}(\nu)$.

Note that any scheme secure under the above definition is also a secure group signature under definition 5.

Construction. The design of a group signature with separated authorities can be based directly on our construction of section 6 with the following modification: the $\mathtt{SETUP}_{\mathsf{GM}}$ procedure will produce $\mathcal{Y}_{\mathsf{GM}} = \langle n, a_0, a, g, h\rangle$ with $\mathcal{S}_{\mathsf{GM}} = \langle p, q\rangle$, whereas the $\mathtt{SETUP}_{\mathsf{OA}}$ will produce $\mathcal{Y}_{\mathsf{OA}} = \langle y, \hat{y}\rangle$ with $\mathcal{S}_{\mathsf{OA}} = \langle x, \hat{x}\rangle$. In all other respects the scheme will proceed in the same fashion. It is straightforward to split the $\mathtt{SETUP}$ procedure to the two authorities, with the condition (as specified in definition 6) that the GM should go first so that the value n is made available; afterwards the OA can select the values $y, \hat{y} \in QR(n)$ with known $\log_g y$ and $\log_g \hat{y}$ and publish the two additional elements to form the combined public key $\mathcal{Y} = \langle n, a_0, a, g, y, \hat{y}\rangle$. To allow the differentiation we specify $\mathcal{Y}_{\mathsf{GM}} = \langle n, a_0, a, g, h\rangle$, $\mathcal{S}_{\mathsf{GM}} = \langle p, q\rangle$, $\mathcal{Y}_{\mathsf{OA}} = \langle y, \hat{y}\rangle$, and $\mathcal{S}_{\mathsf{OA}} = \langle \log_g y, \log_g \hat{y}\rangle$. The design remains unaltered otherwise. In our security proofs we took special care to disassociate the hardness of factoring from anonymity. Taking advantage of this, the following theorem can be easily shown:

Theorem 5. *The group signature with separated authorities presented above is correct and secure; in particular: (i) it is secure against misidentification-attacks under the Strong-RSA assumption in the RO model. (ii) it is secure against framing-attacks under the Discrete-Log hardness assumption over $QR(n)$ with known factorization and the RO model. (iii) it is secure against anonymity-attacks under* DDH-Compo-KF *in the RO model.*

References

1. Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In Lars Knudsen, editor, *Advances in Cryptology – EUROCRYPT ' 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433, Amsterdam, The Netherlands, 2002. Springer.
2. G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In Matthew Franklin, editor, *Financial cryptography: Third International Conference, FC '99, Anguilla, British West Indies, February 22–25, 1999: proceedings*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1999.
3. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO ' 2000*, volume 1880 of *Lecture Notes in Computer Science*. International Association for Cryptologic Research, Springer, 2000.
4. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, Warsaw, Poland, 2003. Springer.

5. Dan Boneh. The decision diffie-hellman problem. In *the Third Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 1998.
6. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO ' 2004*, Lecture Notes in Computer Science. International Association for Cryptologic Research, Springer-Verlag, 2004.
7. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer-Verlag, 2000.
8. Jan Camenisch. Efficient and generalized group signatures. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques*, Lecture Notes in Computer Science, pages 465–479. International Association for Cryptologic Research, Springer, 1997.
9. Jan Camenisch and Anna Lysyanskaya. An identity escrow scheme with appointed verifiers. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO ' 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 388–407. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2001.
10. Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, volume 1514 of *Lecture Notes in Computer Science*, pages 160–174. International Association for Cryptologic Research, Springer-Verlag, 1998.
11. Jan Camenisch and Markus Michels. Separability and efficiency for generic group signature schemes (extended abstract). In Michael j. Wiener, editor, *19th International Advances in Cryptology Conference – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 413–430. Springer, 1999.
12. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. *Lecture Notes in Computer Science*, 1294:410–424, 1997.
13. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology, Proc. of Eurocrypt '91 (Lecture Notes in Computer Science 547)*, pages 257–265. Springer-Verlag, April 1991. Brighton, U.K.
14. L. Chen and T. P. Pedersen. New group signature schemes (extended abstract). In Alfredo De Santis, editor, *Advances in Cryptology—EUROCRYPT 94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1995, 9–12 May 1994.
15. Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, August 2000.
16. 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.
17. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SICOMP*, 30(2):391–437, 2000. A preliminary version appeared in 23rd STOC, 1991.
18. Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proceedings of CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Verlag, 1986.

19. Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368. Springer Verlag, 2001.
20. Oded Goldreich. On the foundations of modern cryptography. In *Proc. 17th Annual International Cryptology Conference – CRYPTO '97*, pages 46–74, 1997.
21. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer Security*, 28:270–299, 1984.
22. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A "paradoxical" solution to the signature problem (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 441–448, Singer Island, Florida, 24–26 October 1984. IEEE.
23. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT ' 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589, Interlaken, Switzerland, 2004. Springer.
24. Aggelos Kiayias and Moti Yung. Extracting group signatures from traitor tracing schemes. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 630–648, Warsaw, Poland, 2003. Springer.
25. Aggelos Kiayias and Moti Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/`.
26. Joe Kilian and Erez Petrank. Identity escrow. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO ' 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 169–185. International Association for Cryptologic Research, Springer, 1998.
27. Kevin S. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 1(2):95–105, 1988.
28. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Baruch Awerbuch, editor, *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 427–437, Baltimore, MY, May 1990. ACM Press.
29. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, March 2000.
30. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO ' 91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1992.

An Analysis of Double Base Number Systems and a Sublinear Scalar Multiplication Algorithm

Mathieu Ciet[1] and Francesco Sica[2,⋆]

[1] Gemplus Security Technologies Department,
La Vigie, ZI Athélia IV, Av. du Jujubier,
B.P. 100, 13705 La Ciotat Cedex, France
mathieu.ciet@gemplus.com
[2] Mount Allison University – AceCrypt,
Department of Mathematics and Computer Science,
67 York Street, Sackville, NB, E4L 1E6, Canada
fsica@mta.ca – http://www.acecrypt.com

Abstract. In this paper we produce a practical and efficient algorithm to find a decomposition of type

$$n = \sum_{i=1}^{k} 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\} \quad \text{with} \quad k \leq \big(c + o(1)\big) \frac{\log n}{\log\log n} \ .$$

It is conjectured that one can take $c = 2$ above. Then this decomposition is refined into an effective scalar multiplication algorithm to compute nP on some supersingular elliptic curves of characteristic 3 with running time bounded by

$$O\left(\frac{\log n}{\log\log n}\right)$$

and essentially no storage. To our knowledge, this is the first instance of a scalar multiplication algorithm that requires $o(\log n)$ curve operations on an elliptic curve over $\mathbb{F}_q$ with $\log q \approx \log n$ and uses comparable storage as in the standard double-and-add algorithm.

This leads to an efficient algorithm very useful for cryptographic protocols based on supersingular curves. This is for example the case of the well-studied (in the past four years) identity based schemes. The method carries over to any supersingular curve of fixed characteristic.

Keywords: Integer decomposition, exponentiation algorithms.

1 Introduction

In asymmetric cryptographic algorithms, the costliest part in execution time is most often the computation of nP, for P belonging to some group[1] denoted

⋆ This work was partially supported by a NSERC Discovery Grant

[1] This operation is called exponentiation or scalar multiplication, according to the multiplicative, respectively additive, notation of the group law.

additively. The standard algorithm to perform this is to decompose n in base 2 and to apply a double-and-add algorithm. If n is randomly chosen with a fixed number of bits, then this algorithm requires on average $\log_2 n$ doublings and $(\log_2 n)/2$ additions. Hence we can say that a classical double-and-add algorithm takes time[2] $\sim c \log n$ (asymptotic to $c \log n$ on average) for some $c > 0$.

In some specific groups, one can improve the constant c by taking for instance signed binary expansions [10] or expansions to other bases [3,11]. We call *linear* an algorithm with running time T satisfying $c \log n < T < d \log n$ for some $c, d > 0$. Similarly, we shall call it *sublinear* if the running time is $o(\log n)$ (little oh) as n goes to infinity. This means that this time is strictly smaller than $\epsilon \log n$ for any $\epsilon > 0$.

The aim of the present work is to present the first practical sublinear scalar multiplication algorithm. The algorithm is sublinear when used on particular supersingular elliptic curves of fixed characteristic (for instance, elliptic curves defined over $\mathbb{F}_p$). We shall deal with characteristic 3 here as an illustration of the method and leave the easy modifications to the interested reader.

The algorithm proceeds as follows. We first decompose the multiplier n as

$$n = \sum_{i=1}^{k} 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\} \tag{1}$$

with $(s_i, t_i) \neq (s_j, t_j)$ for $i \neq j$ and

$$k \leq \bigl(c + o(1)\bigr) \frac{\log n}{\log \log n}$$

for some proven $c > 0$ (conjecturally $c = 2$). This allows us to build a double-and-add algorithm where the number of additions is $O(\frac{\log n}{\log \log n})$.

Then a simple remark allows us to impose a further condition that $\max(s_i) \leq \log^{1-\delta} n$ for any $0 < \delta < 1$. This has the effect of bringing the number of doublings down to $O(\log^{1-\delta} n)$.

We still need to worry about triplings, but on a supersingular curve these are practically as fast a two Frobenius endomorphisms, hence they can be neglected if normal bases are used and even in the case of polynomial bases, they are faster than a doubling or an addition.

Therefore the total running time of our algorithm will be bounded by the number of additions, that is $O(\frac{\log n}{\log \log n})$.

Let us mention that the idea of using (1), called in the literature double base number system, goes back at least to [5], where even an exponentiation algorithm is given [6] with comparable running time as ours. However this algorithm requires $O(\log^2 n)$ in storage, whereas our algorithm only needs a minimal storage. Also [4] develops these ideas into a generic scalar multiplication algorithm that would have fast running time. Unfortunately, no explicit estimate is given at the moment.

[2] The time unit is an elliptic curve operation, say an addition, since a doubling takes a positive proportion of performing an addition.

2 Notations and Mathematical Background

2.1 Notations

We define a s-integer as an integer whose prime factorization contains only the first s primes. A 2-integer will also be called a binumber. An expansion such as (1) will be called a binumber expansion of length k instead of a double base number system, to reflect the fact that the double base is $(2, 3)$.

We are interested in doing asymptotic analysis, at first. We will freely use the abusive notation $f(s) \leq g(s)$ to indicate in fact that for any $C > 1$ one has $f(s) \leq Cg(s)$ as $s \to \infty$. It must then be understood that the optimal asymptotic result will be achieved when we let $C \to 1^+$.

In the same vein, the ϵ's are not always the same from one equation to the next, but have to be understood as an arbitrarily small constant, independent of any other quantity.

Logarithms are taken to the base e in our theoretical analysis, however this is irrelevant in the algorithms, since the ϵ in the exponents absorbs the constants involved in changing bases.

2.2 Continued Fractions

Every real number α has an essentially unique (unique for irrationals) expression as a sequence of "infinite quotients", called continued fraction expansion

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots}}$$

where $(a_s)_{s \geq 0}$ is a sequence of integers which are all positive except $a_0 = \lfloor \alpha \rfloor$. We then write $\alpha = [a_0, a_1, \ldots]$. The a_i's are called partial quotients, and the rationals

$$\frac{p_s}{q_s} = [a_0, a_1, \ldots, a_s]$$

are called the convergents to α. They provide the best rational approximation among all fractions with denominator bounded by q_s. The following recurrence relations are satisfied

$$p_s = a_s p_{s-1} + p_{s-2} \quad \text{and} \quad q_s = a_s q_{s-1} + q_{s-2} \ . \tag{2}$$

It is also known that the convergents hop from one side to the other of α. In other words,

$$\alpha - \frac{p_{2s}}{q_{2s}} > 0 \quad \text{and} \quad \alpha - \frac{p_{2s+1}}{q_{2s+1}} < 0 \tag{3}$$

2.3 Measure of Irrationality

The irrationality measure $\mu(\alpha)$ of $\alpha \in \mathbb{R} - \mathbb{Q}$ is defined as

$$\mu(\alpha) = \sup \left\{ r \in \mathbb{R} \colon \exists \infty \, (p, q) \in \mathbb{Z}^2 \text{ with } \left| \alpha - \frac{p}{q} \right| \leq \frac{1}{q^r} \right\} \ .$$

It is also known that the convergents $\frac{p}{q}$ of the continued fraction expansion of α satisfy

$$\left|\alpha - \frac{p}{q}\right| \leq \frac{1}{q^2} \ , \tag{4}$$

hence $\mu(\alpha) \geq 2$. The set of reals with irrationality measure greater than 2 has Lebesgue measure zero, even if it is equipotent to the reals. Therefore we should expect that a "random" number α would have $\mu(\alpha) = 2$.

2.4 Supersingular Elliptic Curves in Characteristic 3

We refer to [9] for generalities on supersingular elliptic curves. We will focus here on supersingular elliptic curves defined over $\mathbb{F}_{3^p}$ and in particular on the examples E_b given in [1] by the Weierstraß equations

$$y^2 = x^3 - x + b \quad \text{with } b = \pm 1.$$

On these curves, the tripling operation sends $P = (x, y)$ to $3P = (x^9 - b, -y^9)$, so that point tripling is essentially equivalent to two Frobenius applications and can be done in $O(p)$ binary operations using polynomial bases and in $O(1)$ binary operations (rotations) using normal bases. As well, $3^t P$ can be computed in $O(pt)$ binary steps using polynomial bases and essentially $O(1)$ using normal bases. In any case, if $t \leq p$, this cost is less than that of multiplying elements in $\mathbb{F}_{3^p}$ hence is less than one elliptic curve operation. This is the reason why we will neglect this cost henceforth.

3 Short Binumber Expansions

The goal of the present section is to provide an effective algorithm to compute a binumber expansion of any n of length $O(\log n / \log\log n)$, with an explicitly given constant. We will give a mathematical proof with an asymptotic value for the constant, then give a standard conjecture under which one can hope to improve this constant (effective measurements corroborate this – see [4]).

The idea of the algorithm is the use of a more explicit tool than [12], namely a consequence of the following more recent result [7, p.19]. For any $a, b \in \mathbb{N}$ with $(a + b) > e^{1000}$ we have

$$|a \log 3 - b \log 2| \geq \exp\bigl(-153500\bigr) \max(a, b)^{-40499} \tag{5}$$

In particular, this implies that

$$\mu\left(\frac{\log 2}{\log 3}\right) \leq 40500 \tag{6}$$

Let us denote $\mu = \mu(\frac{\log 2}{\log 3})$ and $\frac{p_s}{q_s}$ the sequence of the convergents to $\log_3 2$. Also as before, we let $(a_i)_{i \geq 0}$ be its sequence of partial quotients. Then, choosing any small $\epsilon > 0$, as s becomes large,

$$\frac{1}{q_s^{\mu+\epsilon}} \leq \left| \frac{\log 2}{\log 3} - \frac{p_s}{q_s} \right| < \frac{1}{a_{s+1} q_s^2} \quad . \tag{7}$$

This implies that $a_s \leq q_{s-1}^{\mu-2+\epsilon}$. In view of (2) one has

$$q_s = a_s q_{s-1} + q_{s-2} \leq q_{s-1}^{\mu-1+\epsilon} \tag{8}$$

Let $m > 1$ be a real parameter to be fixed later and let s be henceforth chosen as the smallest even index such that $q_s > m$.

Lemma 1. *Using the above notations we have*

$$\exp\left(\frac{1}{m^{(\mu-1)^3+\epsilon}}\right) < \frac{2^{q_s}}{3^{p_s}} < \exp\left(\frac{1}{m^{\mu-1-\epsilon}}\right) \quad . \tag{9}$$

Proof. By (8) and the definition of s we deduce that

$$q_s \leq m^{(\mu-1)^2+\epsilon} \quad . \tag{10}$$

Together with (3) and the left-hand inequality in (7) we get

$$q_s \frac{\log 2}{\log 3} - p_s > \frac{1}{m^{(\mu-1)^3+\epsilon}}$$

and so

$$q_s \log 2 - p_s \log 3 > \frac{1}{m^{(\mu-1)^3+\epsilon}}$$

which is the left-hand side of (9). The right-hand side is then a consequence of the definition of μ, since

$$q_s \frac{\log 2}{\log 3} - p_s \leq \frac{1}{q_s^{\mu-1-\epsilon}} < \frac{1}{m^{\mu-1-\epsilon}} \quad . \qquad \square$$

Remark 1. Note that in the preceding equation, the ϵ can be dropped by (4) in the probable case that $\mu = 2$.

Theorem 1. *Let n be a large integer. There exists a binumber N satisfying*

$$n - \frac{n}{\log^{\frac{1}{\mu(\mu-1)}-\epsilon} n} \leq N \leq n$$

Remark 2. It is of course possible to give a totally effective version of this theorem, valid for all values of n, but this would clutter the proof with useless technicalities. In practice we see that the range of validity of such inequalities starts at small values of n.

Proof. Our idea is to start from the largest power 3^ν not exceeding n and then to multiply it by the t-th power of $\frac{2^{q_s}}{3^{p_s}}$ which by (9) is very close to 1. However one must be careful of not dividing by too large a power of 3. We will then get a binumber that is very close to n, provided we choose the largest t satisfying both

$$\left(\frac{2^{q_s}}{3^{p_s}}\right)^t \leq \frac{n}{3^\nu} \tag{11}$$

and

$$p_s t < \nu \ . \tag{12}$$

The former inequality, together with (9), implies that

$$t \leq m^{(\mu-1)^3+\epsilon} \tag{13}$$

since $n/3^\nu \leq 3$. Inequality (12) can be automatically verified if m is not too large. Note that (7) and (10) imply that

$$\frac{q_s}{2} < p_s < q_s < m^{(\mu-1)^2+\epsilon} \ . \tag{14}$$

This implies that

$$p_s t \leq m^{(\mu-1)^2+\epsilon} m^{(\mu-1)^3+\epsilon} < \nu = \left\lfloor \frac{\log n}{\log 3} \right\rfloor \ ,$$

if we let

$$m = \left(\log n\right)^{\frac{1}{\mu(\mu-1)^2}-\epsilon} \ . \tag{15}$$

We will henceforth fix m to this value. The choice of t is then dictated by (11) alone and is

$$t = \left\lfloor \frac{\log n - \left\lfloor \frac{\log n}{\log 3} \right\rfloor \log 3}{q_s \log 2 - p_s \log 3} \right\rfloor = \left\lfloor \frac{\log_3 n - \lfloor \log_3 n \rfloor}{q_s \log_3 2 - p_s} \right\rfloor \ . \tag{16}$$

Let $N = 3^\nu \left(\frac{2^{q_s}}{3^{p_s}}\right)^t$. Then N is a binumber less than n and using (9) we get

$$\frac{n}{N} < \exp\left(\frac{1}{m^{\mu-1-\epsilon}}\right) \Longrightarrow N > n\left(1 - \frac{1}{m^{\mu-1-\epsilon}}\right)$$

which in view of (15) proves the left-hand side inequality of the theorem. □

Theorem 2. *Denote the measure of irrationality of* $\log 2/\log 3$ *by* μ. *Then every sufficiently large number* n *can be written as a sum*

$$n = \sum_{i=1}^{k} 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\}$$

with

$$k \leq \mu(\mu-1)\frac{\log n}{\log\log n} + o\left(\frac{\log n}{\log\log n}\right) \ .$$

Proof. Apply Theorem 1 iteratively, starting from $\mathbf{n} = n$, finding N and replacing $\mathbf{n}$ with $\mathbf{n} - N$. At each step, the size of $\mathbf{n}$ is divided by a factor $\log^{\frac{1}{\mu(\mu-1)}-\epsilon} \mathbf{n}$. Therefore, κ steps at most are needed to shrink the size of $\mathbf{n}$ from n to $\sqrt{n}$, where

$$\left(\log^{\frac{1}{\mu(\mu-1)}-\epsilon} \sqrt{n}\right)^{\kappa} = \sqrt{n} \ .$$

This yields

$$\kappa = \left(\frac{\mu(\mu-1)}{2} + \epsilon\right) \frac{\log n}{\log\log\sqrt{n}} \ .$$

Repeating this process, replacing n by $\sqrt{n}$, we see that the number of iterations needed to get down from n to e^4 is bounded by

$$\left(\frac{\mu(\mu-1)}{2} + \epsilon\right) \sum_{0 \le u \le \frac{\log\log n}{\log 2} - 2} \frac{\log \sqrt[2^u]{n}}{\log\log \sqrt[2^{u+1}]{n}} =$$
$$\left(\frac{\mu(\mu-1)}{2} + \epsilon\right) \sum_{0 \le u \le \frac{\log\log n}{\log 2} - 2} \frac{1}{2^u} \frac{\log n}{\log\log n - (u+1)\log 2} \ .$$

Upon splitting the last sum into subsums for $0 \le u < \frac{3\log\log\log n}{\log 2}$ and $\frac{3\log\log\log n}{\log 2} \le u \le \frac{\log\log n}{\log 2} - 2$ we get on the former sum a bound of

$$\left(\frac{\mu(\mu-1)}{2} + \epsilon\right) \log n \sum_{u=0}^{\infty} \frac{1}{2^u} \frac{1}{\log\log n - 4\log\log\log n}$$
$$\le \big(\mu(\mu-1) + \epsilon\big) \frac{\log n}{\log\log n} \ ,$$

while on the latter a smaller bound

$$O\left(\frac{\log n}{2^{\frac{3\log\log\log n}{\log 2}}} \log\log n\right) = O\left(\frac{\log n}{(\log\log n)^2}\right) \ .$$

This proves the theorem.

$\square$

Remark 3. Note that in the preceding theorem one can assume $2^{s_i}3^{t_i} \neq 2^{s_j}3^{t_j}$ if $i \neq j$. In fact by (12) we make sure N in Theorem 1 will not be a pure power of 2, as soon as n is sufficiently large. Below this threshold we can represent any number in base 2 as a sum of $O(1)$ powers of 2, so it suffices to show that $2^{s_{i+1}}3^{t_{i+1}} < 2^{s_i}3^{t_i}$ when these numbers are large. Let $\mathbf{n}$ then be so large that $\log^{\frac{1}{\mu(\mu-1)}-\epsilon} \mathbf{n} > 2$. Following the constructive proof of the preceding theorem, we have

$$2^{s_{i+1}}3^{t_{i+1}} \le \mathbf{n} - 2^{s_i}3^{t_i} \le \frac{\mathbf{n}}{\log^{\frac{1}{\mu(\mu-1)}-\epsilon} \mathbf{n}} < \mathbf{n} - \frac{\mathbf{n}}{\log^{\frac{1}{\mu(\mu-1)}-\epsilon} \mathbf{n}} \le 2^{s_i}3^{t_i} \ .$$

In practice $\mathbf{n}$ does not have to be large, since it is expected that $\mu = 2$.

Input: An integer $n \geq 1$
Output: A set $\mathcal{S} = \{(s_1, t_1), \dots, (s_k, t_k)\}$ with the property that $n = \sum_{i=1}^{k} 2^{s_i} 3^{t_i}$ with s_i, t_i nonnegative integers

```
1.  $i \leftarrow 0$, $\mathcal{S} = \emptyset$
2.  While $n > 0$ do
3.    $i \leftarrow i + 1$
4.    $\nu \leftarrow \lfloor \log_3 n \rfloor$
5.    $m \leftarrow \lfloor \sqrt[3]{\nu} \rfloor$
6.    $s \leftarrow \min\{2j : q_{2j} > m\}$
7.    $q \leftarrow q_s$, $p \leftarrow p_s$
8.    $t \leftarrow \left\lfloor \frac{\log_3 n - \lfloor \log_3 n \rfloor}{q \log_3 2 - p} \right\rfloor$
9.    $N \leftarrow 2^{qt} 3^{\nu - pt}$
10.   $\mathcal{S} \leftarrow \mathcal{S} \cup \{(qt, \nu - pt)\}$, $n \leftarrow n - N$
11. Return $\mathcal{S}$
```

Algorithm 1. Binumber Chain

The proofs of Theorem 1 and 2 show the workings of an algorithm that allows us to find a binumber expansion of n, given as Algorithm 1. The choice for m in step 5 was done to fix ideas (we also assume in implementations that $\mu = 2$) and because this algorithm is to be used in conjunction with Algorithm 2, where some extra conditions, which are satisfied here, are imposed on the binumber expansion.

Note that the storage requirement is minimal and only depends on the size of the size of n. If storing all p_s, q_s with $\max_s(p_s, q_s) \leq M$, we need only $O(\log^2 M)$ bits, since (2) implies $2q_{s-2} < q_s$ and similarly for p_s. Hence we would need only $O\big((\log\log n)^2\big)$ bits of storage.

4 Scalar Multiplication on Supersingular Elliptic Curves in Characteristic 3

It appears that the decomposition described above leads quite naturally in some cases to sublinear multiplication algorithms, that is to algorithms employing $o(p)$ elliptic curve operations to compute nP on an elliptic curve defined over $\mathbb{F}_q$ with $q = 2^p, 3^p$. Of these two types, ternary curves (defined over $\mathbb{F}_{3^p}$) seem to be of some interest in conjunction with recent works on pairing-based cryptographic protocols [2,8]. It happens that examples of supersingular curves over finite fields of characteristic 3 exhibit the right ingredient to make our multiplication algorithm run provably fast. Namely, such elliptic curves possess an extremely fast multiplication by 3 (comparable to the Frobenius endomorphism).

Our previous algorithm does not immediately generalize to a fast multiplication algorithm, since the cost of the latter is expressed in terms of curve additions and doublings/triplings. Theorem 2 results in a sublinear number of curve additions, but, if a doubling or a tripling is performed in a non-negligible fraction of the time cost of an addition, then the overall cost of the scalar multiplication is still comparable to $\log n$. In the case of a supersingular elliptic curve in characteristic 3, we will however prove the following result.

Theorem 3. *Let E be a supersingular elliptic curve defined over $\mathbb{F}_3$ and $P \in E(\mathbb{F}_{3^m})$. Then there exists an algorithm which, on input $n < \#E(\mathbb{F}_{3^m})$, will compute nP in time*

$$O\left(\frac{\log n}{\log\log n}\right) \ .$$

Proof. Let us decompose n as in the proof of Theorem 1. Then (13) and (14) imply that

$$q_s t \leq \log^{1-\delta} n \tag{17}$$

as soon as

$$m = \left(\log n\right)^{\frac{1}{\mu(\mu-1)^2}-\varepsilon}$$

for some fixed $\varepsilon > 0$. Note that in Algorithm 1 our choice for m leads to (17) with $\delta = 1/3$. But $q_s t$ defines an exponent s_i at each step of Algorithm 1. The conclusion is that we can efficiently decompose

$$n = \sum_{i=1}^{I} 2^{s_i} 3^{t_i}, \quad s_i, t_i \in \mathbb{N} \cup \{0\} \tag{18}$$

with

$$I \leq \mu(\mu-1)\,\frac{\log n}{\log\log n} + o\left(\frac{\log n}{\log\log n}\right) \tag{19a}$$

and the additional conditions (see Remark 3)

$$(s_i, t_i) \neq (s_j, t_j) \ , \quad i \neq j \tag{19b}$$

and

$$\max_i s_i \leq \log^{1-\delta} n \ . \tag{19c}$$

From a decomposition with these properties, it is now easy to build a scalar multiplication algorithm that takes asymptotically (as we take $\delta \to 0$) at most $\big(\mu(\mu-1) + o(1)\big)\frac{\log n}{\log\log n}$ curve operations (additions or doublings) to compute nP, as described below.

□

Note that (19b) implies that we can rewrite (18) as[3]

$$n = \sum_{i=1}^{\mathfrak{I}} 2^{s_i} \sum_{j=1}^{\mathfrak{J}_i} 3^{t_{i,j}} \tag{20}$$

where

$$\sum_{i=1}^{\mathfrak{I}} \sum_{j=1}^{\mathfrak{J}_i} 1 = I \quad , \quad s_i > s_{i+1} \quad \text{and} \quad t_{i,j} > t_{i,j+1} \ .$$

This, together with (19c), implies that

$$\mathfrak{I} \leq \log^{1-\delta} n \ . \tag{21}$$

Algorithm 2 describes a practical use of (20) in a scalar multiplication algorithm employing two nested loops, one (internal) on the index j, another (external) on the index i. The total cost of the algorithm is bounded by

$$\sum_{i=1}^{\mathfrak{I}} \sum_{j=1}^{\mathfrak{J}_i} \text{additions} + \mathfrak{I} \text{ doublings} = O\left(\frac{\log n}{\log\log n}\right) \text{ elliptic curve operations}$$

using (19a) and (21). Note that we can neglect the cost of performing $3^{t_{i,j}-t_{i,j+1}}R$, following the discussion in Section 2.4: if we use normal bases, this is clear since the cost of the triplings in steps 5 and 10 is negligible. If we use a suitable polynomial basis then the cost of triplings from those steps is $O(pt_{i,1} + pt_{\mathfrak{I},1}) = O(p^2)$ binary operations. Since this is less than a curve operation, the total cost of triplings is, after (21), $O(\log^{1-\delta} n)$ elliptic curve operations, so that this is negligible with respect to the total number of additions.

5 Conclusion and Ideas for Further Research

We have presented a scalar multiplication algorithm on supersingular elliptic curves (in characteristic 3, easily extendible to other small characteristics) which is sublinear in the length of the multiplier. Its relative speedup over double-and-add or even triple-and-add in cryptographic algorithms making use of these elliptic curves increases with the size of the parameters towards 100%. This means that larger parameter size will result in comparatively more and more advantageous performance with this sublinear algorithm.

From another viewpoint, this sheds new light on the balance between weakened security on such curves and added performance: in increasing the size of the parameters, one would still retain a high performance. In conclusion choosing these curves (with sufficient large parameters for good security) in pairing-based cryptography seems a viable choice that needs further investigation.

[3] The s_i's here are a subsequence of the original s_i's.

Input: A point P on the supersingular elliptic curve E_b and a sequence of pairs of exponents $(s_i, t_{i,j})$ as in (20).
Output: The point Q on the elliptic curve such that $Q = nP$.

```
1.  $Q \leftarrow \mathcal{O}$
2.  For $i = 1$ to $\mathfrak{I} - 1$
3.      $R \leftarrow P$
4.      For $j = 1$ to $\mathfrak{J}_i$
5.          $R \leftarrow 3^{t_{i,j} - t_{i,j+1}} R + P$
6.          $Q \leftarrow Q + R$
7.      $Q \leftarrow 2^{s_i - s_{i+1}} Q$
8.  $R \leftarrow P$
9.  For $j = 1$ to $\mathfrak{J}_{\mathfrak{I}}$
10.     $R \leftarrow 3^{t_{\mathfrak{I},j} - t_{\mathfrak{I},j+1}} R + P$
11.     $Q \leftarrow Q + R$
12. Return $Q$
```

Algorithm 2. Sublinear Multiplication

References

1. P. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *Advances in Cryptology - Proceedings of CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–369. Springer, 2002.
2. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.
3. M. Ciet, T. Lange, F. Sica, and J-J. Quisquater. Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms. In E. Biham, editor, *Advances in Cryptology - Proceedings of Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 388–400. Springer, 2003.
4. V. S. Dimitrov, L. Imbert, and P. K. Mishra. Fast elliptic curve point multiplication using double-base chains. Cryptology ePrint Archive, Report 2005/069, 2005. `http://eprint.iacr.org/`.
5. V. S. Dimitrov, G. A. Jullien, and W. C. Miller. Theory and applications for a double-base number system. In *IEEE Symposium on Computer Arithmetic*, pages 44–53, 1997.
6. V. S. Dimitrov, G. A. Jullien, and W. C. Miller. An algorithm for modular exponentiation. *Information Processing Letters*, 66(3):155–159, 1998.
7. N. Gouillon. *Minorations explicites de formes linéaires en deux logarithmes*. PhD thesis, Université de la Méditerranée Aix-Marseille II, Faculté des Sciences de Luminy, 2003.
8. A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithmic Number Theory, 4th International Symposium, ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

9. A.J. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.
10. F. Morain and J. Olivos. Speeding up the Computations on an Elliptic Curve using Addition-Subtraction Chains. *Inform. Theor. Appl.*, 24:531–543, 1990.
11. J. A. Solinas. An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. In Burton S. Kaliski Jr., editor,*Advances in Cryptology - Proceedings of CRYPTO 1997*, volume 1294 of*Lecture Notes in Computer Science*, pages 357–371. Springer, 1997.
12. R. Tijdeman. On the maximal distance between integers composed of small primes. *Comp. Mathematica*, 28:159–162, 1974.

Power Analysis by Exploiting Chosen Message and Internal Collisions – Vulnerability of Checking Mechanism for RSA-Decryption*

Sung-Ming Yen[1], Wei-Chih Lien[1], SangJae Moon[2], and JaeCheol Ha[3]

[1] Laboratory of Cryptography and Information Security (LCIS),
Dept of Computer Science and Information Engineering,
National Central University, Chung-Li, Taiwan 320, R.O.C.
{yensm, cs222058}@csie.ncu.edu.tw
http://www.csie.ncu.edu.tw/~yensm/

[2] School of Electronic and Electrical Engineering,
Kyungpook National University,
Taegu, Korea 702-701
sjmoon@ee.knu.ac.kr

[3] Dept of Computer and Information,
Korea Nazarene University, Choong Nam, Korea 330-718
jcha@kornu.ac.kr

Abstract. In this paper, we will point out a new side-channel vulnerability of cryptosystems implementation based on BRIP or square-multiply-always algorithm by exploiting specially chosen input message of order two. A recently published countermeasure, BRIP, against conventional simple power analysis (SPA) and differential power analysis (DPA) will be shown to be vulnerable to the proposed SPA in this paper. Another well known SPA countermeasure, the square-multiply-always algorithm, will also be shown to be vulnerable to this new attack. Further extension of the proposed attack is possible to develop more powerful attacks.

Keywords: Chosen-message attack, Cryptography, Side-channel attack, Simple power analysis (SPA), Smart card.

1 Introduction

During the past few years many research results have been published on considering smart card side-channel attacks because of the popular usage of smart cards on implementing cryptosystems. This new branch of cryptanalysis is usually called the *side-channel attack* (SCA).

The power analysis attack is an important category of SCA originally published by Kocher [1] in which both simple power analysis (SPA) and differential power analysis (DPA) were considered. SPA tries to extract the private key by

* This work was supported by University IT Research Center Project.

observing on a single or a very few number of power consumption traces collected from the smart card. DPA consists in performing a statistical analysis of many power consumption traces (say a few thousands or more) of the same algorithm with different inputs.

Exponentiation and its analogy, point scalar multiplication on elliptic curve, are of central importance in modern cryptosystems implementation as they are of the basic operation of almost all modern public-key cryptosystems, e.g., the RSA system [2] and the elliptic curve cryptography [3,4]. Therefore, many side-channel attacks and also the related countermeasures on implementing exponentiation and point scalar multiplication have been reported in the literature.

Some recent works of power analysis attack, e.g., refined power analysis (RPA) [5], zero-value point attack (ZPA) [6], and doubling attack [7], threaten most existing countermeasures for implementing exponentiation and point scalar multiplication, e.g., some countermeasures in [8]. Recently, Mamiya *et al* proposed an enhanced countermeasure which was claimed to resist against RPA, ZPA, classical DPA and SPA, and also doubling attack by introducing a new random blinding technique and also exploiting a well known regular program execution trick (say the square-multiply-always like approach) for each loop iteration.

The main contribution of this paper is that a new SPA by exploiting specific chosen message is proposed in which collecting a single power trace is sufficient to mount a successful attack. An important result obtained is that both the well known SPA resistant countermeasure by using the square-multiply-always algorithm [8] and also the recent and enhanced BRIP algorithm [9] are shown to be vulnerable to this new attack. Further extension on the attack is also pointed out by selecting more general and random input messages which makes the detection of a specific message employed in the basic attack be infeasible and this leads to a more powerful extended attack. Furthermore, the proposed attack is also applicable to implementation of RSA with CRT speedup. Another important observation is that cryptographic padding (e.g., RSA-OAEP [10,11]) is not always useful against simple power attack.

2 Preliminary and Related Works

In this paper, we consider the problem of computing modular exponentiation. In the context of RSA private computation (for example, generating a digital signature or ciphertext decryption), we consider the computation of $S = M^d \bmod n$ where M, d, and n are the input message, the private key, and the modulus integer, respectively.

2.1 Exponentiation Algorithm

Let $\sum_{i=0}^{m-1} d_i \, 2^i$ be the binary expansion of exponent d. The computation $S = M^d \bmod n$ needs efficient exponentiation algorithms to speedup its implementation.

Although numerous exponentiation algorithms have been developed for computing $M^d \bmod n$, practical solutions for devices with constraint computation and storage capabilities (e.g., smart cards) are usually restricted to the basic square-multiply algorithm (refer to Fig. 1 for the left-to-right/MSB-to-LSB version) and some slightly modified ones. The exponentiation algorithm in Fig. 1 processes the bits of the exponent d from the most significant bit (MSB) towards the least significant bit (LSB). An LSB-to-MSB counterpart of the algorithm in Fig. 1 can be available from most related literature.

```
    INPUT: $M, d = (d_{m-1} \cdots d_0)_2, n$
    OUTPUT: $M^d \bmod n$
01  $T = 1$
02  for $i$ from $(m-1)$ downto $0$ do
03    $T = T^2 \bmod n$
04    if $(d_i = 1)$ then $T = T \times M \bmod n$
05  return $T$
```

Fig. 1. Classical left-to-right exponentiation algorithm

2.2 Side-Channel Attacks and Countermeasures

Side-channel attacks are developed based on the fact that in most real implementations some side-channel information (e.g., timing or power consumption) will depend on the instructions being executed and/or the data being manipulated. Therefore, the side-channel information may be exploited to mount a successful attack to retrieve the embedded private key, e.g., the private exponent d in $M^d \bmod n$.

The classical binary exponentiation algorithm in Fig. 1 includes a conditional branch (i.e., the Step (04)) that is driven by the secret data d_i. If the two possible branches behave differently (or the branch decision operation itself behaves distinguishably), then some side-channel analysis (e.g., the simple power analysis–SPA) may be employed to retrieve the secret data d_i. So, further enhancement on the algorithm is necessary.

A novel idea of introducing dummy operations and eliminating secret data dependent statements was proposed previously to enhance the basic algorithms such that the improved versions behave more regularly. Some square-multiply-always (or its counterpart called the double-add-always for point scalar multiplication) based algorithms were already developed (refer to the well known one in Fig. 2 [8] and a recent improvement in Fig. 3 [9]) by employing this observation.

2.3 Doubling Attack

The doubling attack [7] (or called squaring attack for the scenario of exponentiation) is an SPA-based attack which works on the left-to-right square-multiply-always countermeasure (see Fig. 2). The main idea is simply to choose two

```
INPUT: $M, d = (d_{m-1} \cdots d_0)_2, n$
OUTPUT: $M^d \bmod n$
01 $T = 1$
02 for $i$ from $(m-1)$ downto 0 do
03     $T_0 = T^2 \bmod n$
04     $T_1 = T_0 \times M \bmod n$
05     $T = T_{d_i}$
06 return $T$
```

Fig. 2. (SPA protected) Square-multiply-always countermeasure

```
INPUT: $M, d = (d_{m-1} \cdots d_0)_2, n$
OUTPUT: $M^d \bmod n$
01 select a random integer $R$
02 $T_0 = R; T_1 = R^{-1} \bmod n; T_2 = M \times R^{-1} \bmod n$
03 for $i$ from $(m-1)$ downto 0 do
04     $T_0 = T_0^2 \bmod n$
05     if $(d_i = 0)$ then $T_0 = T_0 \times T_1 \bmod n$
06     else $T_0 = T_0 \times T_2 \bmod n$
07 return $T_0 \times T_1 \bmod n$
```

Fig. 3. BRIP countermeasure for exponentiation

strongly related inputs M and M^2 (so being a chosen-message attack) and to observe the collision of two computations for $M^{2(2x+d_i)} \bmod n$ and $M^{4x} \bmod n$ if $d_i = 0$. In the doubling attack, even if the attacker cannot decide whether a computation being performed is squaring or multiplication, the attacker can still detect collision of two operations (basically the squaring operation) within two related computations. More precisely, for two computations $A^2 \bmod n$ and $B^2 \bmod n$, even if the attack cannot tell the values of A and/or B, however the attacker can detect the collision if $A = B$.

The following example given in Table 1 provides the details of the doubling attack. Let the private exponent d be $79 = (1,0,0,1,1,1,1)_2$ and the two related input messages be M and M^2, respectively. The computational process of raising M^d and $(M^2)^d$ using the left-to-right square-multiply-always algorithm reveals the fact that if $d_i = 0$, then both the first computations (both are squarings) of iteration[1] $i-1$ for M^d and iteration i for $(M^2)^d$ will be exactly the same. So, observing collisions within computation on two collected power consumption traces enables the attacker to identify all private key bits of zero value.

The assumption made (was claimed in [7] to be correct experimentally) is very reasonable since the target computations usually take many machine clock cycles and depend greatly on the operands, so the collision is more easy to detect.

[1] Here, the iteration number is denoted decreasingly from $m-1$ downward toward zero.

Table 1. Computations of M^d and $(M^2)^d$ in the square-multiply-always algorithm

i	d_i	Process of M^d	Process of $(M^2)^d$
6	1	1^2 $1 \times M$	1^2 $1 \times M^2$
5	0	M^2 $M^2 \times M$	$(M^2)^2$ $M^4 \times M^2$
4	0	$(M^2)^2$ $M^4 \times M$	$(M^4)^2$ $M^8 \times M^2$
3	1	$(M^4)^2$ $M^8 \times M$	$(M^8)^2$ $M^{16} \times M^2$
2	1	$(M^9)^2$ $M^{18} \times M$	$(M^{18})^2$ $M^{36} \times M^2$
1	1	$(M^{19})^2$ $M^{38} \times M$	$(M^{38})^2$ $M^{76} \times M^2$
0	1	$(M^{39})^2$ $M^{78} \times M$	$(M^{78})^2$ $M^{156} \times M^2$
Return		M^{79}	M^{158}

To protect against the above doubling attack, the random message blinding (RMB) technique should be employed. The RMB technique blinds the original message M to $M \times R \bmod n$ before being signed with a random mask R, and removes a blinding factor $(R^d)^{-1} \bmod n$ from the result to obtain the signature S by computing $M^d = (M \times R)^d \times (R^d)^{-1} \bmod n$.

However, it has been shown in [7] that a regular (in order to be efficient) mask updating, e.g., by $R_i = R_{i-1}^2 \bmod n$ mentioned in [8], might be vulnerable to the doubling attack. So, it was suggested that a real random masking can be employed to avoid the attack.

2.4 The BRIP Countermeasure

Randomized exponentiation algorithms were recently considered as effective countermeasures against DPA by introducing randomization onto the input message or into the computational process of the algorithm in order to remove correlation between the private key and the collected power traces. One of such countermeasures is the BRIP algorithm [9] shown in Fig. 3 (it means binary expansion with random initial point/value) in which the input RSA message is blinded by multiplying with a random integer $R^{-1} \bmod n$.

It was claimed in [9] that the BRIP algorithm can be secure against SPA[2] since there will always be two operations in each iteration, i.e., the Step (04) and one of either the Step (05) or the Step (06).

[2] Of course, the version given in Fig. 3 needs some slight modification (mostly on using well known register indexing trick) to make it be truly SPA resistant. However, this version is sufficient for demonstration purpose.

Remarks. BRIP was originally proposed for ECC context to protect against RPA which requires an inversion for the computation. However, BRIP's authors say that their algorithm can also be applied in $\mathbb{Z}_n$ for cryptosystems based on integer factorization or discrete logarithm. In fact, BRIP can also work efficiently for the above systems if an efficient and secure (against related side-channel attacks, especially the doubling-like attack) random message blinding update process for $\{R_i, R_i^{-1}\}$ can be developed.

3 The Proposed Attack

In the following, an SPA by exploiting chosen input data will be pointed out which is generic and can be extended to some related attacks that will be described in this paper.

3.1 Attack Assumption

The assumption made in this paper is basically the same as what considered in the doubling attack [7] and that in an attack reported in [12]. The validity and practicality of the employed attack assumption was claimed in [7] to be correct by experiment[3].

The assumption is that an adversary can distinguish collision of power trace segments (within a single or more power traces) when the smart card performs twice the same operation even if the adversary is not able to tell which exact computation is done.

Examples of collision instances to be distinguished include modular squaring and modular multiplication. For example, an adversary is assumed to be able to detect the collision of $A^2 \bmod n$ and $B^2 \bmod n$ if $A = B$ even though A and B are unknown.

3.2 Attack on the Square-Multiply-Always Algorithm

In the context of RSA system, given the modulus n, we observed that $(n-1)^2 \equiv 1 \pmod{n}$. This observation can be extended to obtain $(n-1)^j \equiv 1 \pmod{n}$ for any even integer j and $(n-1)^k \equiv n-1 \pmod{n}$ for any odd integer k.

Given $M = n-1$, the square-multiply-always exponentiation algorithm in Fig.2 will have $T = (n-1)^{(d_{m-1}\cdots d_i)_2} \bmod n$ after the Step (05) of iteration i. If $T = 1$, then $(d_{m-1}\cdots d_i)_2$ is an even integer and $d_i = 0$. Otherwise, $T = n-1$ and $(d_{m-1}\cdots d_i)_2$ is an odd integer and $d_i = 1$.

By observing on a *single* collected power trace of performing the algorithm in Fig.2, the attacker can try to identify the value of T (it can only be either 1 or $n-1$) at the end of each iteration and to conduct the aforementioned derivation of each d_i. The approach used to identify the value of d_i is by SPA shown below. Given the two possible values of T at the end of iteration i, there will be only

[3] So, we did not perform another experiment.

two possible computations of the iteration $(i-1)$ shown below and which can be identified by using SPA. In the following statements, the symbol $x \to y$ means that the result of computation x will be assigned to the register y.

– if $d_i = 0$, Step (03) of the iteration $(i-1)$ performs:
  $1^2 \bmod n \to T_0$;
– if $d_i = 1$, Step (03) of the iteration $(i-1)$ performs:
  $(n-1)^2 \bmod n \to T_0$.

Notice that there are only two possible candidate computations of the Step (03). So, during the attack, it does not need to know exactly which of the two observed power consumption patterns of the Step (03) matches with the computation of $(n-1)^2 \bmod n$ (or $1^2 \bmod n$). Only two possible private keys d's will be derived and a trial-and-error approach can be used to select the correct d among the two possibilities. For example, if the MSB of d is presumed to be one, then one of the two possible d's can be selected easily.

All the private key bits can be derived except d_0 by the above SPA. However, d_0 can be known by detecting whether the final result is $T = 1$. On the other hand, in the context of RSA, d_0 is always binary one. Notice that this new SPA is much easier to mount than in the case of a conventional one (to attack the algorithm in Fig. 1) since now the square-multiply-always algorithm performs regularly such that each iteration has one modular squaring followed by a modular multiplication. Therefore, given a collected power trace, it would be much easier to identify the beginning and the end[4] of all iterations and this benefits the proposed new SPA.

Interestingly, a countermeasure originally developed to be resistant to SPA is unfortunately more vulnerable to a new SPA which is much easier to mount compared to the conventional SPA.

3.3 Attack on the BRIP Algorithm

It is interesting to note that the randomized version of square-multiply-always exponentiation in Fig. 3, the BRIP algorithm, is also vulnerable to the above proposed SPA. In the BRIP countermeasure, given $M = n-1$, we observe that at the end of iteration i:

– if $d_i = 0$: after the Step (05), $T_0 = (n-1)^{(d_{m-1}\cdots d_i)_2} \times R = R \bmod n$,
– otherwise if $d_i = 1$: after the Step (06), $T_0 = (n-1)^{(d_{m-1}\cdots d_i)_2} \times R = (n-1) \times R \bmod n$.

Based on the proposed chosen-message SPA, by observing on a *single* collected power trace, the attacker can try to identify the value of T_0 at the end of each iteration i in order to derive d_i. Given the two possible values of T_0 at the end of iteration i, there will be only two possible computations (shown below) of the iteration $(i-1)$ which can be identified by using SPA.

[4] Actually, the computation of the second part (say the Step (04)) of each iteration under the proposed chosen-message SPA are the same, i.e., $1 \times (n-1) \bmod n$. This collision helps to identify the end of each iteration.

- if $d_i = 0$, Step (04) of the iteration $(i-1)$ performs:
  $R^2 \bmod n \rightarrow T_0$;
- if $d_i = 1$, Step (04) of the iteration $(i-1)$ performs:
  $((n-1) \times R)^2 \bmod n \rightarrow T_0$.

In the above approach of SPA, all the private key bits can be derived except d_0. Similarly, d_0 can usually be obtained easily by some other approaches, and for RSA the value of d_0 is known to be one.

It is very important to notice that no matter what the value of R will be, the proposed SPA is applicable. Evidently, the initial random message blinding technique (at the Step (02)) proposed in the BRIP is not resistant against the proposed attack.

Another approach to mount the chosen-message SPA on the BRIP is possible and is shown below. Since there are only two possible input values of T_0 (either R or $(n-1)R \bmod n$) at the beginning of each iteration i, it is always true that $T_0 = R^2 \bmod n$ when finishing the Step (04). After that, one of the two possible modular *multiplications* (Step (05) or Step (06)) will be performed depending on the value of d_i.

- if $d_i = 0$, Step (05) of the iteration i performs:
  $R^2 \times R^{-1} \bmod n \rightarrow T_0$;
- if $d_i = 1$, Step (06) of the iteration i performs:
  $R^2 \times ((n-1)R^{-1}) \bmod n \rightarrow T_0$.

The same that the above attack is applicable no matter what the value of R will be. Hence, the initial random message blinding technique at the Step (02) is still in vain.

3.4 Applicability of the Attack

The proposed attack is applicable to the cases where element of order 2 exists or in any case $(-1)^k$ is computed in prime-order cases. So, the proposed attack now works against

- traditional textbook RSA decryption and signature
- RSA-OAEP decryption [10,11]
- ElGamal decryption [13]

when they are implemented based on the square-multiply-always or the BRIP algorithms. An interesting and important point to observe is that cryptographic padding (e.g., currently used RSA-OAEP) is not always useful against simple power attack. However, the proposed attack does work on RSA-OAEP decryption since the ciphertext validity checking is performed after the RSA private exponentiation computation. So, the attacker still can collect the necessary power trace(s).

But, the proposed attack does not work against the following systems

- most discrete logarithm based signature schemes
- elliptic curve discrete logarithm based decryption and signature (since ECC is usually implemented on the prime-order elliptic curves and there is no element with order 2)
- RSA signature with hash function and/or cryptographic padding.

4 Extension of the Proposed Attack

4.1 Extension to RSA with CRT

In the RSA cryptosystem, let the public modulus be $n = p \times q$ which is the product of two secret prime integers p and q each with roughly $|n|/2$ bits[5]. Prime factorization of the public modulus n can totally break the system.

The well known Chinese Remainder Theorem (CRT) technique [14,15] can be used to speedup the RSA private computation extensively, e.g., the RSA signature computation $S = M^d \bmod n$. In the RSA with CRT, we compute $S_p = M_p^{d_p} \bmod p$ and $S_q = M_q^{d_q} \bmod q$, where $M_p = M \bmod p$, $M_q = M \bmod q$, $d_p = d \bmod (p-1)$, and $d_q = d \bmod (q-1)$. Finally, the signature is computed by using the following Gauss's [14, p.68] (or other more efficient alternative) recombination algorithm

$$S = (S_p \times q \times (q^{-1} \bmod p) + S_q \times p \times (p^{-1} \bmod q)) \bmod n$$

where both $q^{-1} \bmod p$ and $p^{-1} \bmod q$ can be precomputed.

Basically, RSA with CRT is about four times faster than the straightforward approach to compute S directly in terms of bit operations. This CRT-based speedup for RSA private computation has been widely adopted in most systems, especially for implementations with smart card. In the following, we will show that the proposed SPA with chosen message is generic and can be applicable to the RSA with CRT even if the adversary does not know the secret prime integers p and q in advance.

In the attack, the adversary tries to derive d_p (or d_q) during the computation of $M_p^{d_p} \bmod p$ (or $M_q^{d_q} \bmod q$). The adversary provides the chosen message $M = n-1$ to the smart card and observes on the computation of $S_p = M_p^{d_p} \bmod p$ under the implementation of left-to-right square-multiply-always algorithm where $M_p = (n-1) \bmod p$.

We observed that $(n-1)^2 \equiv 1 \pmod{n}$ leads to $M_p^2 \equiv 1 \pmod{p}$ (and $M_q^2 \equiv 1 \pmod{q}$) where $M_p = (n-1) \bmod p$ (and $M_q = (n-1) \bmod p$). The above observation can be extended to obtain $M_p^j \equiv 1 \pmod{p}$ for any even integer j and $M_p^k \equiv M_p \pmod{p}$ for any odd integer k.

The square-multiply-always algorithm in Fig.2 will have $T = M_p^{(d_{p,\ell-1} \cdots d_{p,i})_2} \bmod p$ (where $\ell = |d_p|$) after the Step (05) of iteration i. If $T = 1$, then $(d_{p,\ell-1} \cdots d_{p,i})_2$ is an even integer and $d_{p,i} = 0$. Otherwise, $T = M_p$ and

[5] The symbol $|n|$ means the number of bits of binary representation of n.

$(d_{p,\ell-1} \cdots d_{p,i})_2$ is an odd integer and $d_{p,i} = 1$. By observing on a single collected power trace, the adversary can try to identify the value of T (in order to derive the value of $d_{p,i}$) by SPA. The analysis is summarized in the following.

– if $d_{p,i} = 0$, Step (03) of the iteration $(i-1)$ performs:
  $1^2 \bmod p \rightarrow T_0$;
– if $d_{p,i} = 1$, Step (03) of the iteration $(i-1)$ performs:
  $M_p^2 \bmod n \rightarrow T_0$.

Notice that in the proposed attack the adversary does not need to know the value of $M_p = (n-1) \bmod p$. Recall that p is unknown to the adversary. All the private key bits of d_p can be derived except $d_{p,0}$ by the above SPA. However, $d_{p,0}$ is binary one in the usual case of RSA parameters selection in which d is an odd integer and $p-1$ is an even integer.

Notice also that the proposed attack is still applicable to the RSA with CRT speedup if the BRIP exponentiation algorithm will be employed. The details of this attack can be obtained similarly, so the analysis is omitted here.

It was already well known that given the *partial* private key d_p (or d_q) and the public parameters n and e, both the factorization of n and also the private key d can be available directly. The approach to factorize n is given below. Randomly select an integer X and computes $Y = X^e \bmod n$. Evidently, $Y^d \equiv X \pmod{n}$ and this leads to $Y^{d_p} \equiv X \pmod{p}$ or equivalently $Y^{d_p} - X \equiv 0 \pmod{p}$. With the knowledge of d_p obtained by the proposed SPA, the adversary can derive p by computing

$$p = \gcd(Y^{d_p} - X, n) = \gcd(Y^{d_p} - X \bmod n, n).$$

With p and q, the RSA private key d can be computed. So, the adversary does not need to analyze on the computation of $M_q^{d_q} \bmod q$ in order to derive d_q.

4.2 Extension to Randomly Chosen-Message Attack

An important question to answer about the proposed attack is that whether identification of $n-1$ as input message can be a sufficient countermeasure. Basically, for some cases, the answer is negative because of the following extended and more powerful attack with slightly different assumption. In the extended attack, two power consumption traces are necessary, but the related input messages are far from a fixed and specific value of $n-1$.

The extended attack on the square-multiply-always algorithm (refer to Fig. 2) performs as follows. The adversary selects a pair of input messages M_1 (can be any random message) and $M_2 = M_1 \times (n-1) \bmod n$ and collects the two related power consumption traces of computing $M_1^d \bmod n$ and $M_2^d \bmod n$. The adversary can mount successfully an SPA by observing the collision of middle results between these two power consumption traces in order to identify zero bits of the private key d. The basic idea is that collision will happen when

$$M_1^k \equiv M_2^k \pmod{n}$$

if the exponent k is an even integer.

In the SPA-protected exponentiation algorithm in Fig. 2, if some key bit d_i is zero, then collision on values of T among the two collected power consumption traces can be detected at the end of iteration i since

$$M_2^{(d_{m-1}\cdots d_i)_2} \equiv M_1^{(d_{m-1}\cdots d_i)_2} \cdot (n-1)^{(d_{m-1}\cdots d_i)_2} \equiv M_1^{(d_{m-1}\cdots d_i)_2} \pmod{n}.$$

Therefore, the Step (03) of iteration $(i-1)$ will be a collision instance (the same operation with same operand). This is summarized in the following.

– if $d_i = 0$, Step (03) of the iteration $(i-1)$ of both the two observed computations perform the same:
  $(M_1^{(d_{m-1}\cdots d_i)_2})^2 \bmod n \rightarrow T_0$;
– if $d_i = 1$, Step (03) of the iteration $(i-1)$ of both the two observed computations perform differently:
  either $(M_1^{(d_{m-1}\cdots d_i)_2})^2 \bmod n \rightarrow T_0$
  or $((n-1) \times M_1^{(d_{m-1}\cdots d_i)_2})^2 \bmod n \rightarrow T_0$.

This SPA enables the adversary to derive all the private key bits of d except d_0.

It is interesting to notice that the above extended attack can be considered as a variant of the doubling attack [7]. Both attacks exploit two related chosen messages (however with different forms) and collision detection by SPA on squaring operations (the Step (03)) within two exponentiations.

There is however some difference between the two attacks. In the doubling attack, the adversary observes on collision occurred in two "different" iterations of two power consumption traces. On the contrary, in the proposed extended attack, the adversary observes on collision occurred in the "same" iteration (in fact, the exactly same timing duration) of two power consumption traces. ¿From practical point of view for the SPA scenario, the collision detection on the *same* iteration will be more or less easier than on *different* iterations. This is especially the case when both attacks deal with random input messages, and try to observe on collision of random computations that will be varying/different during the whole process of the algorithm.

One possibility to detect the proposed extended attack is to check the relationship[6] between two input messages on whether $M_j = M_i \times (n-1) \bmod n$ for every pair of i and j. It is however extremely difficult and infeasible to detect all the relationship of input messages since M_i and M_j may not be two consecutive input messages to mount the attack. By the way, it is infeasible to store all previous input messages in order to perform the detection, especially for the applications with smart card.

5 Conclusions

It was previously believed that the BRIP algorithm can be an effective countermeasure against SPA and DPA. However, our research reveals that the BRIP

[6] In the doubling attack, similar approach is to check whether $M_j = M_i^2 \bmod n$ for every pair of i and j.

algorithm is vulnerable to a new SPA. By the way, the well known left-to-right square-multiply-always algorithm (for SPA resistance) is also shown to be insecure against the proposed attack.

Notice especially that the proposed SPA can also be applicable to the scenario of RSA decryption even if RSA-OAEP padding will be considered. The reason is that the ciphertext validity checking is performed after the private exponentiation computation. So, the attacker still can collect the power trace(s).

Some extensions of the attack are also considered in this paper which include the application on RSA with CRT speedup and how to use randomly chosen messages to mount a similar attack.

Acknowledgment

The authors would like to thank Dr. Atsuko Miyaji, Dr. Shiho Moriai, and several anonymous referees for their professional and in-depth suggestions and comments on both technical and editing issues. These suggestions improve extensively to the final version of this paper.

References

1. P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," *Advances in Cryptology – CRYPTO '99*, LNCS 1666, pp. 388–397, Springer-Verlag, 1999.
2. R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystem," *Commun. of ACM*, vol. 21, no. 2, pp. 120–126, 1978.
3. V. Miller, "Uses of elliptic curve in cryptography," *Advances in Cryptology – CRYPTO '85*, LNCS 218, pp. 417-426, Springer-Verlag, 1985.
4. N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, Jan. 1987.
5. L. Goubin, "A refined power-analysis attack on elliptic curve cryptosystems," *Proc. of Public Key Cryptography – PKC '03*, LNCS 2567, pp. 199–210, Springer-Verlag, 2003.
6. T. Akishita and T. Takagi, "Zero-value point attacks on elliptic curve cryptosystem," *Proc. of Information Security Conference – ISC '03*, LNCS 2851, pp. 218–233, Springer-Verlag, 2003.
7. P.-A. Fouque and F. Valette, "The doubling attack – why upwards is better than downwards," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '03*, LNCS 2779, pp. 269–280, Springer-Verlag, 2003.
8. J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 292-302, Springer-Verlag, 1999.
9. H. Mamiya, A. Miyaji, and H. Morimoto, "Efficient countermeasures against RPA, DPA, and SPA," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '04*, LNCS 3156, pp. 343–356, Springer-Verlag, 2004.
10. PKCS #1 v2.1, "RSA Cryptography Standard", 5 January 2001. `http://www.rsasecurity.com/rsalabs/pkcs/`

11. M. Bellare and P. Rogaway, "Optimal asymmetric encryption padding – How to encrypt with RSA," *Advances in Cryptology – EUROCRYPT '94*, LNCS 950, pp. 92–111, Springer-Verlag, 1995.
12. K. Schramm, T. Wollinger, and C. Paar, "A new class of collision attacks and its application to DES," *Proc. of Fast Software Encryption – FSE '03*, LNCS 2887, pp. 206–222, Springer-Verlag, 2003.
13. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985.
14. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997.
15. J.-J. Quisquater and C. Couvreur, "Fast decipherment algorithm for RSA public key cryptosystem," *Electronics Letters*, vol. 18, no. 21, pp. 905–907, 1982.

Optimization of the MOVA Undeniable Signature Scheme

Jean Monnerat[1,⋆], Yvonne Anne Oswald[2], and Serge Vaudenay[1]

[1] EPFL, Switzerland
http://lasecwww.epfl.ch
[2] ETH Zürich, Switzerland

Abstract. This article presents optimization results on the MOVA undeniable signature scheme presented last year by Monnerat and Vaudenay at PKC '04 as well as its generalization proposed at Asiacrypt '04 which is based on a secret group homomorphism. The original MOVA scheme uses characters on $\mathbf{Z}_n^*$ and some additional candidate homomorphisms were proposed with its generalization. We give an overview of the expected performance of the MOVA scheme depending on the group homomorphism. Our optimizations focus on the quartic residue symbol and a homomorphism based on the computation of a discrete logarithm in a hidden subgroup of $\mathbf{Z}_n^*$. We demonstrate that the latter provides a signature generation which is three times faster than RSA.

Keywords: Undeniable signatures, optimization.

1 Introduction

Undeniable signatures, which have been introduced by Chaum and van Antwerpen in [1], differ from classical digital signatures in the verification process. Contrary to classical digital signatures, where anyone holding the public key of the signer is able to verify whether a given signature is valid or not, one has to interact with the signer to be convinced of the validity or the invalidity of the signature. This interaction enables the signer to control the distribution of the signature verification. An undeniable signature scheme therefore consists of a key setup algorithm and a signature generation algorithm, as well as an interactive verification protocol. This protocol is composed of a confirmation and a denial protocol which allow to prove the validity resp. the invalidity of the signature.

In March 2004, a new undeniable signature scheme called MOVA was proposed by Monnerat and Vaudenay [12]. More recently, the same authors generalized this scheme to the more general framework of group homomorphisms [13]. Namely, the MOVA scheme can be seen as the particular case where the underlying homomorphism is a character on $\mathbf{Z}_n^*$. When the choice of the homomorphism is adequate (as for MOVA), this signature scheme allows signatures to be arbitrarily short (typically around 20–30 bits), depending on the required security level.

⋆ Supported by a grant of the Swiss National Science Foundation, 200021-101453/1.

The goal of this paper is to optimize the signature generation algorithm of the generalized scheme based on group homomorphisms and to present a comparison of the signature generation efficiency between the group homomorphisms considered as potential candidates. In particular, we focus on the optimization of characters of order 4 which requires to deal with algorithms computing the quartic residue symbol. Moreover, one quartic residue symbol variant is of particular interest since it is the only homomorphism presenting the special property of having two levels of secret. We propose an application of this property where a delegate of a company needs to sign some pre-agreement of a transaction which will be finalized later by the company using an additional level of secret. We also analyze the case of a homomorphism proposed in [13] consisting of sending elements of $\mathbf{Z}_n^*$ to a cyclic subgroup followed by the computation of a discrete logarithm. We give details on an implementation using a precomputed table of discrete logarithms. A comparison with practical parameters (e.g., a modulus n of 1024 bits) with the Jacobi symbol as well as RSA using standard efficient methods is presented at the end of this article. Our implementations are done in C using the large numbers library GMP [6].

2 The MOVA Scheme

For the sake of simplicity, the generalized scheme [13] will be called MOVA as well. Below we review the main ideas and the signature generation algorithm of this undeniable signature scheme.

First, let us recall some basic definitions from [13] related to the interpolation of group homomorphisms.

Definition 1. *Let G and H be two Abelian groups.*

1. *Given $S := \{(x_1, y_1), \ldots, (x_s, y_s)\} \subseteq G \times H$, we say that the set of points S* interpolates in a group homomorphism *if there exists a group homomorphism $f : G \longrightarrow H$ such that $f(x_i) = y_i$ for $i = 1, \ldots, s$.*
2. *We say that a set of points $B \subseteq G \times H$* interpolates in a group homomorphism with another set of points *$A \subseteq G \times H$ if $A \cup B$ interpolates in a group homomorphism.*

The central idea of the generalized MOVA scheme is to consider a secret group homomorphism Hom between two publicly known Abelian groups Xgroup and Ygroup as the signer's secret key. The order of the group Ygroup is public and is denoted as d. The signer then chooses a set Skey $\subseteq$ Xgroup $\times$ Ygroup of Lkey points such that Skey interpolates in a unique homomorphism, namely Hom. The signer chooses Skey $:= \{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$ in varying ways depending on the choice of one of the setup variants presented in [13]. The size of the parameter Lkey depends on the setup variant choice too. Then, to sign a given message m the signer computes Lsig values $\mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_{\mathrm{Lsig}} \in$ Xgroup from m by using a random oracle and computes $\mathrm{Hom}(\mathrm{Xsig}_i) := \mathrm{Ysig}_i$

for $1 \leq i \leq \mathrm{Lsig}$. Finally, the signature of m with respect to the secret key Hom is

$$\sigma := (\mathrm{Ysig}_1, \ldots, \mathrm{Ysig}_{\mathrm{Lsig}}).$$

In the verification step, the verifier will first send the message-signature pair (m, σ) he would like to verify. If the pair is valid, the signer launches a confirmation protocol with the verifier in which he proves that the set of points $\{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$ interpolates in a group homomorphism (namely Hom) with the set $\{(\mathrm{Xsig}_1, \mathrm{Ysig}_1), \ldots, (\mathrm{Xsig}_{\mathrm{Lsig}}, \mathrm{Ysig}_{\mathrm{Lsig}})\}$. Otherwise, the signer launches a denial protocol in which he proves that the set of points $\{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$ does not interpolate in a group homomorphism with the set $\{(\mathrm{Xsig}_1, \mathrm{Ysig}_1), \ldots, (\mathrm{Xsig}_{\mathrm{Lsig}}, \mathrm{Ysig}_{\mathrm{Lsig}})\}$. More details about the confirmation and denial protocols can be found in [13].

We state the so-called "Group Homomorphism Interpolation Problem" introduced in [13].

S-GHI Problem (Group Homomorphism Interpolation Problem)
Parameters: two Abelian groups G and H, a set of s points $S \subseteq G \times H$.
Input: $x \in G$.
Problem: find $y \in H$ such that (x, y) interpolates with S in a group homomorphism.

It is shown in [13] that the resistance of this scheme against existential forgery under a chosen-message attack relies on the hardness of the GHI problem with parameters $G = \mathrm{Xgroup}$, $H = \mathrm{Ygroup}$ and $S = \mathrm{Skey}$. Hence, for a homomorphism (more formally a family of homomorphisms) for which the Skey-GHI problem is hard, we can assume that there is no easier method to forge a signature than performing an (online) exhaustive search. Furthermore, if the homomorphism is such that it is hard to find any information bit on y in the Skey-GHI problem, the security level against an existential forgery attack depends exactly on the signature size which is $\mathrm{Lsig} \cdot \log_2(d)$.

3 Homomorphisms

In this section, we briefly describe some instances of the group homomorphism Hom considered in [13] such as characters on $\mathbf{Z}_n^*$ [12], the RSA encryption homomorphism [5,14] or the discrete logarithm in a hidden subgroup [13].

3.1 Characters on $\mathbf{Z}_n^*$

Definition 2. *Let n be an integer. A character χ on $\mathbf{Z}_n^*$ is a group homomorphism from $\mathbf{Z}_n^*$ to $\mathbf{C} - \{0\}$ i.e.,*

$$\chi(ab) = \chi(a)\chi(b) \textit{ for all } a, b \in \mathbf{Z}_n^*.$$

The characters on $\mathbf{Z}_n^*$ form a group with respect to the composition of functions. The order of a character χ is its order with respect to the group of characters. It is important to note that a character of order d maps any element to a dth root of the unity. In the MOVA scheme, the study focused on the characters of order 2, 3 and 4. In this article, we will not consider the case $d = 3$ since the algorithmic issues are similar to the case $d = 4$.

For more details about characters, we refer to the article on the MOVA scheme [12] and the textbook of Ireland and Rosen [7].

Jacobi Symbol We consider a public modulus $n = pq$ where p, q are two large secret primes. From the theory of characters, it directly follows that there exist exactly 4 characters of order 2 on $\mathbf{Z}_n^*$, namely the Jacobi symbols $(\cdot/n)_2$, $(\cdot/p)_2$, $(\cdot/q)_2$ and the trivial character. Note that the Jacobi symbol $(\cdot/n)_2$ and the trivial character are not suitable for our purpose since they can be efficiently computed without the knowledge of the factorization of n.

Quartic Characters The theory of the characters of order 4 naturally occurs in the context of Gaussian integers. We recall the required background related to our study. Most of these results are taken from [7].

The ring of the Gaussian integers is defined as

$$\mathbf{Z}[i] := \{a + bi \mid a, b \in \mathbf{Z}\}.$$

The norm of an element $\alpha = a + bi$ is defined as $N(\alpha) = \alpha \cdot \bar{\alpha} = a^2 + b^2$, where $\bar{\alpha}$ denotes the complex conjugate of α. $\mathbf{Z}[i]$ is well known to be Euclidean which implies that we can talk about the gcd of two Gaussian integers and there is an Euclidean division: given α, $\beta \in \mathbf{Z}[i]$ with $\beta \neq 0$, there exists γ, $\delta \in \mathbf{Z}[i]$ s.t. $\alpha = \gamma\beta + \delta$ and $N(\delta) < N(\beta)$. Note that γ and δ are not necessarily unique. The units (invertible elements) of $\mathbf{Z}[i]$ are ± 1, $\pm i$. We say that two elements α, β are associate if and only if $\alpha = u\beta$ for a unit u. The gcd of two Gaussian integers is uniquely defined up to an associate. Moreover, we say that two Gaussian integers α and β are relatively prime iff the only common divisors are units, i.e., their gcd is a unit. In this case we will use the notation $\gcd(\alpha, \beta) \sim 1$. Any prime element of $\mathbf{Z}[i]$ is of the following form or the associate of an element of this form:

1. $1 + i$
2. $q \equiv 3 \pmod 4$ a prime in $\mathbf{Z}$
3. π such that $N(\pi) \equiv 1 \pmod 4$ is a prime in $\mathbf{Z}$

Any Gaussian integer has a unique decomposition into primes up to a unit. For any prime $\sigma \in \mathbf{Z}[i]$, the quotient $\mathbf{Z}[i]/(\sigma)$ is a field with $N(\sigma)$ element. This allows to define the *quartic residue symbol.*

Definition 3. *Let α, $\beta \in \mathbf{Z}[i]$ be such that $(1 + i) \nmid \beta$ and $\gcd(\beta, \alpha) \sim 1$. The* quartic residue symbol *is defined as $\chi_\beta : \mathbf{Z}[i] \to \{\pm 1, \pm i\}$*

$$\chi_\beta(\alpha) = \begin{cases} u \text{ such that } \alpha^{\frac{N(\beta)-1}{4}} \equiv u \pmod{\beta}, u \in \{\pm 1, \pm i\}, \text{ if } \beta \text{ is prime} \\ \prod_i \chi_{\beta_i}(\alpha), \text{ if } \beta = \prod_i \beta_i, \ \beta_i \text{ prime} \end{cases}$$

The quartic residue symbols which are considered for MOVA [12] are chosen as follows. Let p, q be two rational primes such that $p \equiv q \equiv 1 \pmod 4$. There exist π, σ such that $p = \pi\bar{\pi}$ and $q = \sigma\bar{\sigma}$. π and σ can be computed with the help of the algorithms of Tonelli and Cornacchia (for more details see [3]). Then, we choose $\mathrm{Hom} = \chi_\beta$ with $\beta = \pi$ or $\beta = \pi\sigma$. Moreover, we take $\mathrm{Xgroup} := \mathbf{Z}_n^*$ which is a natural choice since $\mathbf{Z}[i]/(\pi\sigma) \simeq \mathbf{Z}_n$ and $\mathbf{Z}[i]/(\pi) \simeq \mathbf{Z}_p$.

From the properties of the quartic residue symbol and the Jacobi symbol, we can show that $(\chi_{\pi\sigma}(a))^2 = (a/n)_2$ for any $a \in \mathbf{Z}$. Therefore, without the knowledge of the factorization of n we can easily deduce one bit of $\chi_{\pi\sigma}(a)$. In practice, we will compress this quartic residue symbol to one bit sending $1, i$ to the bit 0 and $-1, -i$ to the bit 1. To decompress, it suffices to compute the Jacobi symbol to retrieve the right quartic residue symbol. Hence, with this quartic residue symbol we have to perform two times more evaluations than with χ_π for the same level of security against an existential forgery. This shows that the signature generation will be anyway less efficient for $\chi_{\pi\sigma}$ than for χ_π.

A motivation for using $\chi_{\pi\sigma}$ is, that this character has two levels of secret, namely the secret key $\pi\sigma$ does not allow to factorize n. As mentioned in [13] an expert group knowledge of the group $\mathbf{Z}_n^*$ is required in order to convert a signature into an ordinary one. Here, this expert group knowledge corresponds to the ability to factorize. Hence, we can imagine an application where a mobile delegate of a company is able to sign some pre-agreement of some contracts or transactions using $\chi_{\pi\sigma}$ which can be confirmed by a server of the company. Later, the delegate sends a report to his company, which then can issue an ordinary signature for a final agreement by converting the signature of the delegate. In such a scenario, even if the delegate loses his key or it is stolen, he can contact his company before a confirmation of the signature is performed. In any case, the company never converts a signature before it is convinced that the delegate key was not lost or stolen.

3.2 RSA

Following the long tradition of the RSA based cryptography, an undeniable signature scheme based on RSA [14] was proposed in 1997 by Gennaro et al.[5]. This scheme can be seen as a special case of the generalized MOVA scheme when the homomorphism is the RSA encryption function defined on a modulus of safe primes. So, the signature is generated as for the regular RSA signature scheme.

3.3 Discrete Logarithm in a Hidden Subgroup

Another homomorphism suitable for the generalized MOVA scheme is based on the discrete logarithm in a hidden subgroup.

Let n be such that $n = pq$ with $p = rd + 1$, q, d prime, $\gcd(q-1, d) = 1$, $\gcd(r, d) = 1$ and g generating a subgroup of $\mathbf{Z}_p^*$. We obtain g by choosing a random element $h \in \mathbf{Z}_n^*$ until h satisfies $h^r \bmod p \neq 1$ and we set $g = h^r \bmod p$. Like this we find a homomorphism by "sending" the input in a hidden cyclic subgroup of order d and then computing its discrete logarithm with respect to the generator g,

$$\begin{aligned} \phi : \mathbf{Z}_n^* &\longrightarrow \mathbf{Z}_d \\ x &\longmapsto \log_g(x^r \bmod p). \end{aligned}$$

4 Quartic Residue Symbol

4.1 Background

We recall some properties of the quartic residue symbol which play a crucial role in the algorithms we will consider. To this end, we introduce the notion of "primarity".

We say that a Gaussian integer $\alpha = a + bi$ is primary if and only if either $a \equiv 1 \pmod 4$, $b \equiv 0 \pmod 4$ or $a \equiv 3 \pmod 4$, $b \equiv 2 \pmod 4$. It can be shown that for any nonunit $\alpha \in \mathbf{Z}[i]$ with $(1+i) \nmid \alpha$, there is a unique associate of α which is primary.

Theorem 4. *Let $\beta = a + bi$, α and α' be some Gaussian integers such that $\gcd(\beta, \alpha) \sim \gcd(\beta, \alpha') \sim 1$ and $(1+i) \nmid \beta$. The following properties hold.*

1. Modularity: *If $\alpha \equiv \alpha' \pmod \beta$ then $\chi_\beta(\alpha) = \chi_\beta(\alpha')$.*
2. Multiplicativity: *$\chi_\beta(\alpha\alpha') = \chi_\beta(\alpha)\chi_\beta(\alpha')$.*
3. Quartic Reciprocity Law: *If α and β are primary,*

$$\chi_\alpha(\beta) = \chi_\beta(\alpha) \cdot (-1)^{\frac{N(\alpha)-1}{4} \cdot \frac{N(\beta)-1}{4}}.$$

4. Complementary Reciprocity Laws: *If β is primary,*

$$\chi_\beta(i) = i^{\frac{N(\beta)-1}{4}} \text{ and } \chi_\beta(1+i) = i^{\frac{a-b-b^2-1}{4}}.$$

4.2 Basic Algorithm

Description To compute the quartic residue symbol $\chi_\beta(\alpha)$ directly, one has to know the factorization of β into primes over $\mathbf{Z}[i]$ and the computation contains an exponentiation. To avoid this factorization as well as the costly exponentiation we apply the properties of the quartic residue symbol iteratively. First we reduce α to an element $\hat{\alpha}$ equivalent to α modulo β and that satisfies $N(\hat{\alpha}) < N(\beta)$. From now on, such a reduction of an element α modulo β will be denoted $\mathrm{Red}_\beta(\alpha)$. Note that the obtained $\hat{\alpha} \leftarrow \mathrm{Red}_\beta(\alpha)$ is not necessarily unique. Then, we find the unique representation $\hat{\alpha} = i^j \cdot (1+i)^k \cdot \alpha'$ with α' primary and employ the multiplicativity property and the complementary laws of the quartic residue symbol. Next, we interchange α and β according to the law of reciprocity and start again. Hence, the size of both α and β decrease progressively. We stop the iteration process when α or β is a unit. The detailed algorithm is described in Algorithm 1.

Algorithm 1 Basic Algorithm Quartic Residuosity in $\mathbf{Z}[i]$

```
Require: $\alpha, \beta \in \mathbf{Z}[i] \setminus \{0\}$, $\gcd(\alpha, \beta) \sim 1$ and $(1+i) \nmid \beta$
Ensure: $c = \chi_\beta(\alpha)$  ($c = 0 \Leftrightarrow \chi_\beta(\alpha)$ is not defined)
 1: $\alpha \leftarrow \mathrm{Red}_\beta(\alpha)$
 2: if $\alpha = 0$ then $c = 0$ end if
 3: let primary $\alpha_1, \beta_1 \in \mathbf{Z}[i]$ be defined by
    $\alpha = (i)^{i_1} \cdot (1+i)^{j_1} \cdot \alpha_1$ and
    $\beta = (i)^{i_2} \cdot \beta_1$
 4: let $m, n \in \mathbf{Z}$ be defined by $\beta_1 = m + ni$
 5: $t \leftarrow \frac{m-n-n^2-1}{4} j_1 + \frac{m^2+n^2-1}{4} i_1 \bmod 4$
 6: replace $\alpha$ with $\beta_1$, $\beta$ with $\alpha_1$
 7: $t \leftarrow t + \frac{(N(\alpha)-1)(N(\beta)-1)}{8} \bmod 4$
 8: while $N(\alpha) > 1$ do
 9:    (LOOP INVARIANT: $\alpha, \beta$ are primary)
10:    $\alpha \leftarrow \mathrm{Red}_\beta(\alpha)$
11:    let primary $\alpha_1$ be defined by $\alpha = (i)^{i_1} \cdot (1+i)^{j_1} \cdot \alpha_1$
12:    let $m, n \in \mathbf{Z}$ be defined by $\beta = m + ni$
13:    $t \leftarrow t + \frac{m-n-n^2-1}{4} j_1 + \frac{m^2+n^2-1}{4} i_1 \bmod 4$
14:    replace $\alpha$ with $\beta, \beta$ with $\alpha_1$
15:    $t \leftarrow t + \frac{(N(\alpha)-1)(N(\beta)-1)}{8} \bmod 4$
16: end while
17: if $N(\alpha) \neq 1$ then $c \leftarrow 0$ else $c \leftarrow i^t$ end if
```

Computation of Related Subfunctions For this algorithm we have to implement a few functions for calculating basic operations in the ring of Gaussian integers (let $\alpha, \beta \in \mathbf{Z}[i]$):

1. Multiplication: $\alpha \cdot \beta$
2. Modular reduction: $\mathrm{Red}_\beta(\alpha)$
3. Norm: $N(\alpha)$
4. Division by $(1+i)^r$
5. Primarisation: transforms α into its primary associate if possible

The multiplication and the norm are trivially implemented by performing integer multiplications between the appropriate integer components.

The division of α by $(1+i)^r$ can be done by first raising $(1+i)$ to the power of r and then dividing α by the result. We propose a way of achieving the same by only using shift operations, additions, and interchanging the imaginary and real part if necessary. The following equations demonstrate our procedure. Let $\alpha = a + bi$,

$$\frac{\alpha}{(1+i)} = \frac{a+b}{2} + \frac{b-a}{2} i,$$

$$\frac{\alpha}{(1+i)^r} = \frac{i^{3k}\left(\frac{a}{2^k} + \frac{b}{2^k} i\right)}{(1+i)^\ell}, \quad r = 2k + \ell.$$

If $r = 2k,\ k \in \mathbf{N}$ we shift the real and the imaginary parts of α by k to the right and multiply them by -1 and/or interchange them depending on the value of $3k$. If r is odd, there is an additional subtraction and addition to perform.

The primarisation function we used consists of a few congruency tests and it also determines the number of times we have to multiply α by i to get the primary associate of α.

The computation of $\mathrm{Red}_\beta(\alpha)$ is done according to [9] using an Euclidean division and rounding appropriately.

To find the representation of α we proceed as follows. First calculate the norm of α, $N(\alpha)$. Then find j maximal such that $2^j \mid N(\alpha)$. Divide α by $(1+i)^j$ and transform the result into its primary associate.

In the implementation of the algorithm we need to ensure $(1+i) \nmid \beta$ and $\gcd(\alpha, \beta) \sim 1$. The first requirement is taken care of by applying the primarisation function on β. If we cannot find a primary associate, β is divisible by $(1+i)$ and we terminate. For the second condition we check in every iteration whether $\mathrm{Red}_\beta(\alpha) \rightarrow 0$. This would imply $\gcd(\alpha, \beta) \not\sim 1$ and we terminate.

4.3 Algorithm of Damgård and Frandsen

Description The most expensive operation used in the algorithm described above is $\mathrm{Red}_\beta(\alpha)$. Damgård and Frandsen present in [2] an efficient algorithm for computing the cubic residue symbol in the ring of Eisenstein integers $\mathbf{Z}[\zeta]$. Their algorithm can be transformed into an algorithm for the quartic residue symbol in the ring of Gaussian integers.

There are three main differences to the basic algorithm. Instead of using $\mathrm{Red}_\beta(\alpha)$ to reduce α, they suggest using $\alpha - \beta$. This takes much less time but increases the number of iterations needed. Furthermore they only interchange α and β, if $N(\alpha) < N(\beta)$. It is not necessary to calculate $N(\cdot)$ exactly for this purpose, an approximation $\tilde{N}(\cdot)$ suffices. They demonstrate how one can compute an approximate norm $\tilde{N}(\alpha)$ in linear time. Instead of adding up the squares of the real and the imaginary part of α, one replaces all but the 8 most significant bits of the real and the imaginary part of α with zeroes and computes the norm of the resulting Gaussian number.

Their algorithm takes $O(\log^2 N(\alpha\beta))$ time to compute $\chi_\beta(\alpha)$.

4.4 Other Algorithms

In addition to the above, we studied papers concerning algorithms for the quartic residue symbol by Weilert. In [17] he presents a fast gcd algorithm for Gaussian integers. Based on this gcd algorithm and using some properties of the Hilbert symbol he demonstrates in [18] how to construct an algorithm for the quartic residue symbol. This algorithm involves calculating an Euclidean descent and storing some intermediate results for later use. This algorithm presents a very fast asymptotic complexity which is even better than that of Damgård and Frandsen.

Algorithm 2 Damgård and Frandsen's Algorithm Quartic Residuosity in $\mathbf{Z}[i]$

```
Require: $\alpha, \beta \in \mathbf{Z}[i] \setminus \{0\}$, $\gcd(\alpha, \beta) \sim 1$ and $(1+i) \nmid \beta$
Ensure: $c = \chi_\beta(\alpha)$  ($c = 0 \Leftrightarrow \chi_\beta(\alpha)$ is not defined)
 1: let primary $\alpha_1, \beta_1 \in \mathbf{Z}[i]$ be defined by
    $\alpha = (i)^{i_1} \cdot (1+i)^{j_1} \cdot \alpha_1$ and
    $\beta = (i)^{i_2} \cdot \beta_1$
 2: let $m, n \in \mathbf{Z}$ be defined by $\beta_1 = m + ni$
 3: $t \leftarrow \frac{m-n-n^2-1}{4} j_1 + \frac{m^2+n^2-1}{4} i_1 \bmod 4$
 4: replace $\alpha$ with $\alpha_1$, $\beta$ with $\beta_1$
 5: if $\tilde{N}(\alpha) < \tilde{N}(\beta)$ then
 6:    interchange $\alpha$ and $\beta$ and adjust $t$
       $t \leftarrow t + \frac{(\tilde{N}(\alpha)-1)(\tilde{N}(\beta)-1)}{8} \bmod 4$
 7: end if
 8: while $\alpha \neq \beta$ do
 9:    (LOOP INVARIANT: $\alpha, \beta$ are primary)
10:    let primary $\alpha_1$ be defined by $\alpha - \beta = (i)^{i_1} \cdot (1+i)^{j_1} \cdot \alpha_1$
11:    let $m, n \in \mathbf{Z}$ be defined by $\beta = m + ni$
12:    $t \leftarrow t + \frac{m-n-n^2-1}{4} j_1 + \frac{m^2+n^2-1}{4} i_1 \bmod 4$
13:    replace $\alpha$ with $\alpha_1$
14:    if $\tilde{N}(\alpha) < \tilde{N}(\beta)$ then
15:       interchange $\alpha$ and $\beta$ and adjust $t$
          $t \leftarrow t + \frac{(\tilde{N}(\alpha)-1)(\tilde{N}(\beta)-1)}{8} \bmod 4$
16:    end if
17: end while
18: if $\alpha \neq 1$ then $c \leftarrow 0$ else $c \leftarrow i^t$ end if
```

However, as mentioned by Damgård et al. in [2], the fastest algorithms for practical inputs in the case of the Jacobi symbol are based on binary gcd algorithms [11]. Weilert proposed a binary gcd algorithm for Gaussian integers in [16] as well, but did not adapt it to the computation of the quartic residue symbol. Algorithms for cubic and quartic residue symbols taking this approach was proposed by Damgård et al. in [2] arguing that this is likely to provide a more efficient algorithm than the asymptotically fast variant of Weilert [18] in practice. Therefore, we have chosen to implement Algorithm 2 which takes this binary approach since we need a fast algorithm for practical inputs rather than the best asymptotic complexity.

5 Discrete Logarithm in a Hidden Subgroup

One suitable homomorphism for the generalized MOVA scheme is the one mentioned in Subsection 3.3. This homomorphism ϕ satisfies $\phi(x) = \log_g(x^r \bmod p)$. It consists of a modular exponentiation followed by a discrete logarithm computation. The modular exponentiation can be implemented by the classical methods such as the square-and-multiply method. For the discrete logarithm computation we consider three variants which are the use of a precomputed table of all discrete logarithms, the Shanks baby-step giant-step (BSGS) algorithm and

Pollard's rho method. The choice of the algorithm will strongly depend on the amount of memory the signer has at disposal, namely the Pollard rho method requires almost no memory while the BSGS method is a time-memory tradeoff. Below we discuss the method of precomputed table and we refer to [10] for a description of the two other methods.

Given p prime, g a generator of a cyclic group G, subgroup of $\mathbf{Z}_p^*$, and $d = |G|$, we construct a table with entries (g^j, j) for $0 \leq j \leq d$. Building this table is a very time and memory consuming task, but once the table exists, finding the discrete logarithm consists of a simple look up operation.

There are several ways of constructing such a table. One can use a two dimensional array and sorting it by the first component. Finding the discrete logarithm is then reduced to a binary search. Alternatively, one can use conventional hashing on the first component to store the entries in a hash table, in which case placing an entry and searching for an entry in the table takes constant time. Another advantage is the fact, that we do not need space for g^i. Especially when $p \gg d$, this can save an enormous amount of memory. The only difficulties are finding a suitable hash function and dealing with collisions without losing too much time.

Time complexity of the construction of the table is $O(d)$ multiplications (plus $O(d \log d)$ comparisons to sort). Space complexity is $O(d(\log d + \log p))$ for the sorted table, resp. $O(d \log d)$ for the hash table. The running time for the sorted table is $O(\log d)$, for the hash table O(1).

6 Implementation

The implementation of all algorithms has been written in C using the GNU Multiple Precision Arithmetic Library (GMP) [6]. This library provides highly optimized arithmetic functions on large numbers. Most of the basic computations in $\mathbf{Z}$ have been performed using GMP such as integer multiplication or the modular exponentiation. For all implemented homomorphisms, we focused on the case where the modulus n is of size of 1024 bits.

6.1 Quartic Residue Symbol

Our principal optimization effort focused on the two algorithms computing the quartic residue symbol. In particular, we minimized the number of function calls, used some of the more sophisticated GMP functions, reduced the number of `mpz_t` (C data type for a multiple precision integer) variables whenever possible and applied general C optimization techniques such as described in [4,8]. In addition, we used profiling and tried out different compiler optimization levels.

The basic algorithm has been implemented using the above remarks as well as the methods for computing the subfunctions which are explained in Subsection 4.2. We proceed in the same way for the algorithm of Damgård and Frandsen. Additionally, we tested whether the use of an approximative norm allows to obtain effective improvements. We implemented both the standard norm and the norm Damgård and Frandsen suggest. The standard norm consists of only two GMP functions: one multiplication and one combined addition/multiplication whereas the approximate norm involves one bit scan to determine the size of the real part, one shift operation to extract the 8 most significant bits, one multiplication for the squaring of these 8 bits and another shift operation to put the result back to its correct position. We apply the same procedure on the imaginary part and we add the two approximate squarings up. In short, we need four additional operations to reduce the size of the numbers we have to multiply.

As GMP is a highly optimized library, computing the standard norm takes little time and the additional operations of the approximate norm only amortise if the real and the imaginary part are larger than 2048 bits. This and the fact that the norm of α and β decreases with each iteration convinced us to use the standard norm instead.

6.2 Discrete Logarithm

Here, we would like to present how we manage the computation of the discrete logarithm in the case of the precomputed table.

In this suggested variant of the generic homomorphic signature scheme, p is typically a 512 bit and d a 20 bit prime. Creating a table with d entries of size 532 bits is impossible on a usual desktop computer. Therefore we decided to use a hash table (key 512 bits, data 20 bits, 2^{20} entries). We found some existing hash table data structures written in C, but they do not fulfill our requirements. They are either too slow, support C types only, do not allow tables that large and/or they store the key as well.

To avoid problems, we did not adapt any of the existing data structures, but implemented a hash table ourselves providing enough storage and a collision handling mechanism suitable for our needs. Our solution is a hash table consisting of an array of unsigned integers. This array is of maximal length (2^{24}) to reduce collisions.

An unsigned integer is 32 bits long, so it was possible to store the data for the logarithm as well as using one of the higher order bits as a flag for collisions. Because the key is large and we wanted to avoid any unnecessary computation, we chose to use the 24 least significant bits of the key as the index into the hash table, in case of collision the next 24 bits, etc. By selecting 24 bits instead of the possible 20 bits, we minimize the occurrence of collisions. Tests have shown that most collisions are resolved by choosing the next 24 bits. We tried out other hash functions, but we did not achieve a gain of speed. This way, the size of the table is 64 MB.

To find the correct discrete logarithm for $y \in G$, one has to check if the collision flag at the corresponding array field is set, to decide if one can return the logarithm stored in the field or if one has to continue with the next field.

The implementation of the BSGS was done in a similar way. As the table contains much less entries, collisions hardly ever occur. The implementation of the Pollard rho method did not require any special treatment.

7 Results

In this section we present the results of the timing measurements we conducted to determine how well the different algorithms perform. In order to measure the running time precisely, we used functionalities offered by `frequence_cpu.h` by Victor Stinner [15]. The tests have been done on an Intel(R)4 1.4 GHz Desktop Computer with 256 MB RAM. Our results are average values produced by test series of 1000 tests.

7.1 Quartic Residue Symbol

We have considered the quartic residue symbol $\chi_\beta(\alpha)$ where α is a Gaussian integer with real and imaginary part of 1024 bits and $\beta = \pi\sigma$ a product of two primes and of size of 512 bits in each component. In such a situation, we have to consider a variant of the Damgård and Frandsen algorithm, we call the mixed algorithm. Namely, since α is much bigger than β it is more efficient in this case to compute first $\hat{\alpha} \leftarrow \mathrm{Red}_\beta(\alpha)$ and apply the Damgård and Frandsen algorithm on $\chi_\beta(\hat{\alpha})$. Timed results and number of iterations are given in Table 1.

Table 1. Quartic Residue Symbol with $\beta = \pi\sigma$

	time in ms	iterations
Basic algorithm	32.12	248.81
Damgård's algorithm	50.63	766.12
Mixed algorithm	24.65	511.92

The mixed algorithm is then the most judicious choice for fast implementations. The same phenomenon occurs for the case $\beta = \pi$ as well.

7.2 Signature Generation

Here, we finally compare the time required for generating a MOVA signature with the different homomorphisms. We consider a signature size of 20 bits. We omit the time required by the generation of the values Xsig_i's. Hence, we just have to compare the time required for computing 20 Jacobi symbols $(\cdot/p)_2$ (or $(\cdot/q)_2$), 20 quartic residue symbols with $\beta = \pi\sigma$, 10 quartic residue symbols with $\beta = \pi$, 1 homomorphism based on the discrete logarithm in a hidden subgroup and 1 RSA homomorphism. We recall that for all these homomorphisms, we take a modulus n of size of 1024 bits. Results are given in Table 2.

We have implemented the Jacobi symbol using a similar algorithm as Algorithm 1 and the basic GMP subroutines in order to have a fair comparison with our implementation of the quartic residue symbol. We note that the highly optimized GMP implementation of the Jacobi symbol `mpz_jacobi` provides the fastest signature generation and that the quartic residue symbol χ_π

Table 2. Results Comparison Signature Schemes

Homomorphism	time in ms
Quartic Residue Symbol ($\beta = \pi\sigma$)	493.01
Quartic Residue Symbol ($\beta = \pi$)	90.32
Jacobi Symbol (ordinary algorithm)	25.22
Jacobi Symbol (`mpz_jacobi`)	2.32
Discrete Logarithm (Precomputed Table)	9.66
Discrete Logarithm (BSGS)	19.47
Discrete Logarithm (Pollard's rho)	74.93
RSA	33.87

is about 4 times slower than our implementation of the Jacobi symbol. This is mainly due to the fact that all operations are performed in $\mathbf{Z}[i]$ instead of $\mathbf{Z}$. We remark that the variant $\chi_{\pi\sigma}$ is much slower than for χ_π since we have to perform two times more quartic residue computations and that β is two times greater. The variants of the discrete logarithm offer a very competitive homomorphism. In particular, except for the variant using the Pollard rho method this homomorphism is even more efficient than RSA. Finally, we can see that a 20-bit MOVA signature can be three times faster than a regular RSA signature.

8 Conclusion

We provided an overview of the implementation of the different candidate homomorphisms for the generalized MOVA scheme. Our principal case study concerned the quartic residue symbol, since the literature dedicated to its implementation is poor. We showed that the signature generation of the most efficient variant of the quartic residue symbol takes less than 4 times our implementation of the Jacobi symbol. The principal reason is that arithmetic operations are performed in $\mathbf{Z}[i]$ which are more costly than in $\mathbf{Z}$. We motivated the use of another variant of quartic residue symbol with two levels of secret by showing an application. This variant is the least efficient homomorphism of our comparison and requires about half a second to perform a signature generation on a classical workstation. When the signer has at least 64 MB memory, we demonstrated that using the homomorphism based on the discrete logarithm gives the most efficient signature generation after the Jacobi symbol implementation of GMP. However, if we take into account the cost of the confirmation protocol, this homomorphism is preferable to the characters. Finally, it is worthwhile to note that this implementation is about three times faster than a regular RSA signature scheme. We provided a clear overview of expected MOVA performance depending on the choice of the group homomorphism. Future work should further consider implementations with protection against side channels.

References

1. D. Chaum and H. van Antwerpen, *Undeniable Signatures*, Advances in Cryptology - Crypto '89, LNCS **435**, pp. 212-217, Springer, 1989.
2. I.B. Damgård and G.S. Frandsen, *Efficient Algorithms for GCD and Cubic Residuosity in the Ring of Eisenstein Integers*, FCT '03, LNCS **2751**, pp. 109-117, Springer, 2003.
3. H. Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics **138**, Springer, 2000.
4. S. Garg, *How to optimize C/C++ Source - Performance Programming*, 2002, `http://bdn.borland.com/article/0,1410,28278,00.html`.
5. R. Gennaro, T. Rabin, and H. Krawczyk, *RSA-Based Undeniable Signatures*, Journal of Cryptology, **13**(4), pp. 397-416, Springer, 2000.
6. The GNU Multiple Precision Arithmetic Library, `http://www.swox.com/gmp/`.
7. K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory: Second Edition*, Graduate Texts in Mathematics **84**, Springer, 1990.
8. M.E. Lee, *Optimization of Computer Programs in C*, 2001, `http://vision.eng.shu.ac.uk/bala/c/c/optimisation/l/optimization.html`.
9. V. Lefèvre, *Entiers de Gauss (sujet d'étude XM')*, 1993, `http://www.vinc17.org/math/index.fr.html`.
10. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
11. S.M. Meyer and J.P. Sorendson, *Efficient Algorithms for Computing the Jacobi Symbol*, Journal of Symbolic Computation **26**(4), pp. 509-523, 1998.
12. J. Monnerat and S. Vaudenay, *Undeniable Signatures Based on Characters: How to Sign with One Bit*, PKC '04, LNCS **2947**, pp. 69-85, Springer, 2004.
13. J. Monnerat and S. Vaudenay, *Generic Homomorphic Undeniable Signatures*, Advances in Cryptology - Asiacrypt '04, LNCS **3329**, pp. 354-371, Springer, 2004.
14. R.L. Rivest, A. Shamir and L.M. Adleman, *A Method for Obtaining Digital Signatures and Public-key Cryptosystem*, Communications of the ACM, vol. **21**(2), pp. 120-126, 1978.
15. V. Stinner, 2003, `frequence_cpu.h`, `frequence_cpu.c`, `http://www.haypocal.com/`.
16. A. Weilert, *(1+i)-ary GCD Computation in Z[i] is an analogue to the Binary GCD Algorithm*, Journal of Symbolic Computation **30**(5), pp. 605-617, 2000.
17. A. Weilert, *Asymptotically fast GCD Computation in Z[i]*, Algorithmic Number Theory, LNCS **1838**, pp. 595-613, Springer, 2000.
18. A. Weilert, *Fast Computation of the Biquadratic Residue Symbol*, Journal of Number Theory **96**, pp. 133-151, 2002.

Questionable Encryption and Its Applications

Adam Young[1] and Moti Yung[2]

[1] LECG LLC*
ayoung@mitre.org
[2] RSA Labs and Columbia University
moti@cs.columbia.edu

Abstract. In this paper we investigate a primitive called a questionable encryption that is related to oblivious transfer. We consider a mobile agent that asymmetrically encrypts plaintext data from the host machine that it resides on and then broadcasts the resulting ciphertext so that it can be obtained by the creator of the agent. We formally define the notion of a *questionable encryption* scheme that can be used to perform this operation. The user of a questionable encryption scheme chooses to generate a real or fake public key. The choice is conveyed to the key generation algorithm which then outputs a poly-sized witness and either a real or fake key pair. If the public key is 'real' then it produces decipherable encryptions and the poly-sized witness proves this. If the key is generated to be 'fake' then it produces indecipherable encryptions (even with the private key) and the poly-sized witness proves this. Without knowledge of the witness it is intractable to distinguish between the two types of public keys. We present a construction for a questionable encryption scheme based on the Paillier cryptosystem. We prove the security of the scheme based on the difficulty of deciding n^{th} degree composite residuosity. When applied to this application, the creator of the agent retains the exclusive ability to reveal whether or not the agent in fact transmits plaintexts. Our results show that agents that appear to compute asymmetric encryptions may in fact not (in a provable sense). We present other applications of questionable encryptions as well.

Keywords: Public key cryptosystem, Paillier cryptosystem, composite residuosity problem, decision composite residuosity problem, semantic security, questionable encryption, deniable encryption, oblivious transfer.

1 Introduction

Mobile agents have been an active area of research and in this paper we investigate a new tool that can be used to enhance the privacy of such agents. Typically, one of two threat models are used when designing modible agents. The first is the *honest but curious* model in which the host machines that the agent traverses are honest enough not to interfere with the operation of the agent, but

* Author is now at MITRE Corporation.

are "curious" about the data it contains and the results of the computations of the agent. The other threat model allows the hosts to be active adversaries that may introduce faults into the computations of the mobile agent.

In this paper we operate under the former threat model and address the following issue. *Does a mobile agent that appears to asymmetrically encrypt data really do so?* One can envision a scenario in which the system operator of a host jumps to the conclusion that the agent encrypts data since it passes a value that appears to be a public key to an asymmetric encryption function.

By dissecting the agent it can be determined that it uses a correct asymmetric cipher implementation. The question then becomes whether or not the public key is properly formed and whether or not the creator of the agent is in possession of the corresponding private key. The creator is not likely to include non-interactive zero-knowledge proofs in the agent for the benefit of proving that the requisite algebraic properties of the public key hold. In this paper we observe that for a variety of public key cryptosystems, the "public key" cannot be immediately construed as such. This has immediate consequences for proving whether or not the agent even transmits host data.

The integrity of public keys gives rise to the following cryptographic problem. Can we devise a plug-in for a well known asymmetric cryptosystem that accepts normal-looking public keys but that produces ciphertexts that provably cannot be decrypted by *anyone*? The answer to this is yes. We formally define a *questionable encryption scheme* that accomplishes this and present an instantiation (plug-in) for the Paillier cryptosystem.

The key generation algorithm of a questionable encryption scheme generates a poly-sized witness and either a real or fake key pair. The user chooses whether to create a real key pair or a fake key pair. If the public key is 'real' then it produces decipherable asymmetric encryptions and the witness proves their decipherability. If the key is 'fake' then it produces indecipherable encryptions and the witness proves that no such ciphertext can be deciphered. Without knowledge of the witness it is intractable to distinguish between the two types of 'public keys.' We call these witnesses of encryption and non-encryption, respectively.

When a mobile agent outputs a questionable encryption it is intractable to determine whether it outputs a valid asymmetric ciphertext or a value that to all intents and purposes is random. This holds even when the actions of the agent are recorded immediately after deployment, and even if the agent is *reverse-engineered* by the system administrator of a host.

We present applications of questionable encryptions in Section 7. Questionable encryptions are related to $(1,2)$-oblivious transfer, all-or-nothing disclosure of secrets, and deniable encryptions. Differences between questionable encryptions and these primitives are given in Appendix A.

2 Questionable Encryptions

In this section we cover basic notation and definitions. Let PTM denote a probabilistic poly-time Turing machine. We let $a \mid b$ denote that integer b is evenly divisible by integer a. When x is a binary string, $|x|$ denotes the number of bits in x. When x is a positive integer it is understood that $|x|$ is the number of bits in the base-2 representation of the integer. A Blum integer is the product of two prime powers p^r and q^s such that $p, q \equiv 3 \ mod \ 4$ with r and s odd. Let $\mathbb{Z}_n^*(+1)$ denote those elements in $\mathbb{Z}_n^*$ with Jacobi symbol 1 with respect to n. Recall that a is a *pseudosquare* mod n provided that a is not a quadratic residue mod n and that $a \in \mathbb{Z}_n^*(+1)$.

Let p_1 be a value between 0 and 1 inclusive. The probabilistic statement below is to be read as follows. The probability that the assertion to the right of the colon holds after the ordered execution of the operations to the left of the colon is less than p_1.

$$Pr[b \in_R \{0,1\}, (x,y) = G_b(1^K) \ : \ A(x,y) = b] < p_1 \tag{1}$$

The following definition is from [13].

Definition 1. *v is negligible if for every constant $c \geq 0$ there exists an integer k_c such that $v(k) < \frac{1}{k^c}$ for all $k \geq k_c$.*

Thus, ν is negligible in k if it vanishes faster than any inverse polynomial in k.

Let k be a security parameter. Define $G_0(\cdot)$ to be a probabilistic poly-time algorithm that on input 1^k outputs a pair of values (x, y). Similarly, define $G_1(\cdot)$ to be a probabilistic poly-time algorithm that on input 1^k outputs a pair of values (x, y). The generator G_1 outputs a private key x and corresponding public key y for the encryption algorithm E and corresponding decryption algorithm D. Let $S_{1,k}$ denote the set of possible outputs of $G_1(1^k)$. Similarly, let $S_{0,k}$ denote the set of possible outputs of $G_0(1^k)$. Let M be the message space for E and let C be the corresponding ciphertext space. Let $c = E(m, y)$ denote the encryption of $m \in M$ under y and let $m = D(c, x)$ denote the corresponding decryption operation.

The notion of security for (G_1, E, D) can be any well-accepted notion of security, e.g. semantic security against known-plaintexts attacks, security against adaptive chosen ciphertext attacks, etc. In the definition below, the notion of security is simply preserved. That is, the questionable encryption scheme is only required to be as secure as the underlying asymmetric cryptosystem. Thus, it is assumed that (G_1, E, D) is *secure* and that it produces *decipherable* encryptions.

Definition 2. *Let (F, G_0, G_1, E, D) be a 5-tuple of algorithms that are public, let $S_{0,k} \cap S_{1,k} = \emptyset$, and let (G_1, E, D) be a* **secure** *asymmetric cryptosystem. If F is an efficiently computable predicate satisfying $F(x, y) = b \Leftrightarrow (x, y) \in S_{b,k}$ for all $(x, y) \in S_{0,k} \cup S_{1,k}$ and,*
(1) [indecipherability] for all PTMs A, for all k, for all $(x, y) \in S_{0,k}$, and for every two messages $m_0, m_1 \in M$,

$$Pr[b \in_R \{0,1\}, c = E(m_b, y) \ : \ A(c, x, y) = b] = \tfrac{1}{2},$$

and,
(2) [indistinguishability] for all PTM A, for all polynomials Q, and for all large enough k,

$$|Pr_{(x,y)\ \in_R\ S_{0,k}}[A(y)=1] - Pr_{(x,y)\ \in_R\ S_{1,k}}[A(y)=1]| < \frac{1}{Q(k)}$$

then (F, G_0, G_1, E, D) *is a* **perfect questionable encryption** *scheme.*

A variant of this is a *computational questionable encryption* scheme in which requirement 1 is weakened as follows.

(1) For all PTMs A there exists a negligible function ν_A such that for all large enough k, for all $(x,y) \in S_{0,k}$, and for any two messages $m_0, m_1 \in M$,

$$Pr[b \in_R \{0,1\}, c = E(m_b, y)\ :\ A(c,x,y) = b] \leq \tfrac{1}{2} + \nu_A(k),$$

This is reminiscent of the notion of message indistinguishability of encryptions. It essentially states that no efficient algorithm A can distinguish between the encryptions of any two messages, even when given $(x,y) \in S_{0,k}$. This requirement is critical since it implies that key pairs generated using G_0 are "no good."

Requirement (2) states that it is intractable to decide whether y is a public key generated using G_1 or a fake public key generated using G_0. This requirement is critical since given requirement (1) it implies that without knowing x it is intractable to decide if y and E produce real encryptions or not.

Observe that (x,y) serves as a witness that $(x,y) \in S_{0,k}$ or $S_{1,k}$. This follows immediately from the fact that $F(x,y) = b \Leftrightarrow (x,y) \in S_{b,k}$. Since y and F are public, the user who generates the pair need only disclose x to prove or disprove whether or not messages can be recovered from c. Hence, a questionable encryption scheme is more general than a normal asymmetric scheme since the user can (1) prove that it encrypts data when $(x,y) \in S_{1,k}$, or (2) prove that it doesn't encrypt data when $(x,y) \in S_{0,k}$.

3 Review of Paillier

In this section we review the Paillier public key cryptosystem [19]. Note that $\lambda(n) = \text{lcm}(p-1, q-1)$ where λ denotes Carmichael's function. Key generation utilizes the function $L(u,n) = \frac{u-1}{n}$.

```
PaillierKeyGeneration($1^k$):
1. generate random $k/2$-bit primes $p$ and $q$
2. compute $n = pq$ and $t = \text{lcm}(p-1, q-1)$
3. choose $g \in_R \mathbb{Z}^*_{n^2}$
4. if $\gcd(L(g^t \bmod n^2), n) \neq 1$ then goto step 3
5. output $((n,g),(p,q))$ and halt
```

The order of g modulo n^2 is v where $v \equiv 0 \bmod n$. To encrypt $m < n$, the value $r \in_R \mathbb{Z}^*_n$ is chosen. The ciphertext is $c = g^m r^n \bmod n^2$. We remark that instead of choosing $r < n$ randomly, the selection $r \in_R \mathbb{Z}^*_n$ is used.[1]

[1] In Section 3 of the original paper [19] this value is indeed selected from $\mathbb{Z}^*_n$ to show that $\mathcal{E}_g$ is bijective.

```
PaillierDecrypt$(c, (n, g), (p, q))$:
Output: plaintext $m < n$
1. compute $t = \mathrm{lcm}(p-1, q-1)$
2. output $m = L(c^t \bmod n^2, n)L(g^t \bmod n^2, n)^{-1} \bmod n$ and halt
```

The Paillier public key cryptosystem is semantically secure against plaintext attacks based on the assumed intractability of solving the decision composite residuosity problem.

4 A Construction Based on Paillier

We will now present our construction that is based on Paillier. The Paillier key generation is altered slightly so that it generates p and q to be safe primes. Recall that p is a safe prime if $(p-1)/2$ is prime. The use of safe primes enables a simple method to generate the fake value g that has order $\lambda(n)$.

```
$G_0(1^k)$:
1. generate random $k/2$-bit safe primes $p$ and $q$
2. compute $n = pq$ and $t = \mathrm{lcm}(p-1, q-1)$
3. choose $g_1 \in_R \mathbb{Z}_{n^2}^*$ and compute $g = g_1^n \bmod n^2$
4. if the order of $g$ is not $t$ then goto step 3
5. output $((n, g), (p, q))$ and halt
```

Since p and q are safe primes the order of g can be computed efficiently.

```
$G_1(1^k)$:
1. generate random $k/2$-bit safe primes $p$ and $q$
2. compute $n = pq$ and $t = \mathrm{lcm}(p-1, q-1)$
3. choose $g \in_R \mathbb{Z}_{n^2}^*$
4. if $(\gcd(L(g^t \bmod n^2), n) \neq 1)$ then goto step 3
5. output $((n, g), (p, q))$ and halt
```

```
$F(p, (n, g))$:
1. if $|p| \geq |n|$ then output failure and halt
2. if $n \ mod \ p \neq 0$ then output failure and halt
3. compute $q = n/p$ and $t = \mathrm{lcm}(p-1, q-1)$
4. if $p$ and $q$ are not $k/2$-bit safe primes then output failure and halt
5. if $g \notin \mathbb{Z}_{n^2}^*$ then output failure and halt
6. if $(\gcd(L(g^t \bmod n^2), n) = 1)$ then output 1 and halt
7. if $g$ has order $t$ then output 0 else output failure and halt
```

To questionably encrypt a large amount of data efficiently, a hybrid scheme can be used. For example, the lower 128 bits of a random plaintext can be used as a symmetric key that is used to encrypt the plaintext. This of course will provide only computational indecipherability, not perfect indecipherability.

We implemented our questionable encryption scheme. See Appendix B for details.

5 Security of the Questionable Encryption Scheme

Fact 1: If $n = pq$ where p and q are primes of equal bit length, then by cycling through all of the values $y \in \mathbb{Z}_n^*$ and computing $y^n \bmod n^2$ all of the n^{th} residues modulo n^2 are generated.

Fact 1 follows implicitly from Lemma 1 in [19] that shows that the function $\mathcal{E}_g(x, y) = g^x y^n \bmod n^2$ is bijective since the n^{th} residues are definable as the image by this function of all pairs $(0, y < n)$.

Theorem 1. *Let $m < n$ be any plaintext message and let $(n, g) \in S_{0,k}$. Then $c = E(m, n, g)$ is unconditionally secure.*

Proof. Consider a computationally unbounded adversary that tries to learn m from c. Recall that the adversary is given p and q (we want to show indecipherability even for the owner of the encryption key pair). From the definition of Paillier encryption it follows that,

(1) $c = g^m r^n \bmod n^2$ where $r \in_R \mathbb{Z}_n^*$

Since $(n, g) \in S_{0,k}$ it follows from the definition of $G_0(1^k)$ that,

(2) g has order $\lambda(n)$

It follows that g^m is an n^{th} residue in (1). Since $r \in_R \mathbb{Z}_n^*$ and since $|p| = |q|$ (i.e., p does not divide $q - 1$ evenly and vice-versa) it follows from Fact 1 that $r^n \bmod n^2$ is an n^{th} residue selected uniformly at random from the set of all n^{th} residues modulo n^2. Therefore c is uniformly distributed among the n^{th} residues modulo n^2. $\diamond$

We will now show what will happen in practice when one tries to decipher c with (p, q) and the fake g. From the definition of Paillier encryption it follows that $c^{\lambda(n)} \equiv (g^{\lambda(n)})^m r^{n\lambda(n)} \bmod n^2$. But Euler's generalization of Fermat's little theorem implies that $c^{\lambda(n)} \equiv (g^{\lambda(n)})^m \bmod n^2$ since $\lambda(n^2) = n\lambda(n)$. Since $(n, g) \in S_{0,k}$ it follows from the definition of $G_0(1^k)$ that g has order $\lambda(n)$. It follows that $c^{\lambda(n)} \bmod n^2 = 1$. From the definition of Paillier decryption it follows that $\mathrm{L}(c^{\lambda(n)} \bmod n^2, n)$ is zero. Therefore, the numerator in the decryption equation is zero and leaves the group entirely.

Theorem 2. *For all PTM A, for all polynomials Q, and for all large enough k,*

$$|Pr_{(p,(n,g)) \in_R S_{0,k}}[A((n, g)) = 1] - Pr_{(p,(n,g)) \in_R S_{1,k}}[A((n, g)) = 1]| < \frac{1}{Q(k)}$$

Proof. Note that the composite n is selected in the same fashion in G_0 and G_1. It remains to consider g. The theorem follows immediately from the definition of computationally indistinguishable probability distributions and the presumed intractability of distinguishing n^{th} residues modulo n^2 from non-residues. $\diamond$

Theorem 3. *If the Paillier asymmetric cryptosystem is secure then the 5-tuple (F, G_0, G_1, E, D) is a perfect questionable encryption scheme.*

6 Other Constructions

Due to space limitations we cannot formally introduce other constructions. However, we give the basic approach for other constructions below.

6.1 Computational Indecipherability

The questionable encryption scheme for RSA [21] uses a public exponent e that is a 160-bit prime. It utilizes primes p and q such that $\gcd(e, p-1) = e$ and $\gcd(e, q-1) = e$. When given (n, e), deciding whether or not $e \mid \phi(n)$ is closely related to the Phi-Hiding problem [9]. When e divides $p-1$ and $q-1$ it is intractable to perform decryption. This follows from the fact that to date, there is no known algorithm for efficiently computing e^{th} roots mod p with such a large prime e. For details see [1]. In addition there are too many roots to check. It follows that the "ciphertexts" that are produced using the fake (p, q) cannot be effectively deciphered.

The ElGamal [12] scheme is as follows. Define $S_{1,k} = \{(x, y) : y = g^x \bmod p$ and $x \in \mathbb{Z}_q\}$. The value x is the witness of encryption. Define $S_{0,k} = \{(x, y) : y = H(x)$ and $x \in \mathbb{Z}_q\}$. Here H is a random function with the same range as the space of public keys. The value x serves as a witness of non-encryption. The intractability assumption is that it is intractable to compute $log_g(H(x)) \bmod p$ for a randomly chosen x. This approach extends to Cramer-Shoup [11].

A construction for Blum-Goldwasser [2] is as follows. Recall that Blum-Goldwasser (BG) is based on Blum-Blum-Shub [3]. In normal BG, the primes satisfy $p \equiv q \equiv 3 \bmod 4$. The fake key pair uses $p \equiv q \equiv 1 \bmod 4$. The basic idea is that for a given bit position in the ciphertext, these "bad" primes make it intractable to disambiguate the correct square root that is a quadratic residue mod pq. The ciphertext is made to be twice as long as normal to ensure that a computationally bounded adversary cannot recover the first half of the Blum-Blum-Shub (BBS) stream. In a forthcomming paper we show that it is infeasible to accurately recover a bit in the first half of the BBS pseudorandom bit stream.

All three of these constructions provide only computational indecipherability. Consider RSA in which the message is padded with a large amount of redundancy. A computationally unbounded adversary could factor n, compute all e^{th} roots and recover the plaintext.

6.2 Perfect Indecipherability

In Goldwasser-Micali (GM) [14] the public key is (n, u) where $n = pq$ is a Blum integer and u is a pseudosquare mod n. The creator of the mobile agent can set u to be a quadratic residue mod n. The fake public key (n, u) is indistinguishable from a real GM public key under the assumed difficulty of distinguishing quadratic residues from pseudosquares mod pq.

The use of a quadratic residue u in GM guarantees that no one can decrypt ciphertexts computed using (n, u). This follows from the fact that the encryption of a binary 1 is unconditionally indistinguishable from the encryption of a binary

0 (both ciphertexts are randomly chosen quadratic residues). With a “fake” u, it follows that private plaintext data is not transmitted outside the host of the mobile agent.

6.3 Why Paillier is Unique

Note that *it is possible, theoretically* that an algorithm exists that can decrypt a questionable encryption produced using one of the schemes in Subsection 6.1. When the GM or Paillier based questionable encryption schemes are used, this argument cannot be made.

The GM based scheme produces large ciphertexts whereas Paillier produces small ciphertexts. This, combined with the fact that Paillier produces perfectly indecipherable ciphertexts is the reason why we chose to focus on the Paillier based scheme.

7 Applications

We present a few applications of questionable encryptions. The primitive is likely to be useful for many problems in which oblivious transfer is already used.

7.1 Application 1: Oblivious Transfers

An application is an agent that conducts a “1-out-of-2” (or n) oblivious transfer. In the case that the encryption scheme is homomorphic and multiplying the encryptions sums up the cleartext, the questionable encryption scheme enables a distributed agent to move around and do the encrypt-and-multiply of the ciphertext which will return the 1 out of 2 sum of a value and nothing about the other value(s). The values can be viewed as a distributed vector. This is applicable in conducting oblivious surveys. In case the vector is of length one this is the usual ”1-out-of-2” Oblivious Transfer, whereas for larger vectors that return the accumulated value it may be called “SUM 1-out-of-2” (or 1-out-of-n) Oblivious Transfer.

Questionable encryptions can also be used in a variant of the above in the area of (distributed) database applications that exhibit a form of private information retrieval. The agent contains within it $n - 1$ fake public keys and one real public key. This is proven non-interactively in the code without showing which keys are real or fake. The agent traverses the network and accumulates plaintexts. Whether or not a given plaintext (or an accumulation of a row of values in relational database terminology) has been acquired is then an uncertainty.

In the above applications, the fact that the means of achieving the Oblivious Transfer is a (homomorphic) encryption schemes is crucial in achieving the distributed accumulation variants.

7.2 Application 2: Agent Privacy

The questionable encryption scheme can be used to ensure the privacy of the agent operation (with respect to whether or not the agent even performs asymmetric encryption). This application is carried out as follows. The creator of a given agent chooses to perform one of the following when designing the agent.

1. (private information transmission) (x, y) is generated using G_1 and y is placed in the agent. The agent is deployed. The agent obtains host data and asymmetrically "encrypts" it using y. The resulting "ciphertexts" are then broadcast (e.g., using an approach in [23]). The creator decrypts the messages that are broadcast by the agent.
2. (non-transmission agent) (x, y) is generated using G_0 and y is placed in the agent. The agent is deployed. The agent obtains host data and asymmetrically "encrypts" it using y. The resulting "ciphertexts" are then broadcast The creator cannot use x to decrypt the messages that are broadcast by the agent. The creator can at any time post the witness of non-encryption x to a public bulletin board after the agent has been studied. It will then be clear that $F(x, y) = 0$.

Consider the possibility that multiple creators create multiple mobile agents. When some agents perform private information transmission and some are non-transmission agents, then there is no way to tell the difference between the cases unless one or more witnesses are revealed. Furthermore, if a few witnesses of non-encryption are revealed, then this will set a concrete *precedent* that mobile agents do not necessarily transmit host data outside the host, even though they might appear to.

A direct use of a deniable encryption scheme cannot be used to construct such an agent. To see this consider the following attempt to use a deniable encryption scheme for this.

Recall that a deniable encryption scheme requires that the sender and receiver share a secret key so that the receiver can identify which of the possible plaintexts is the correct one. The creator places this key within the mobile agent. The agent is deployed and it broadcasts deniable encryptions. If the creator retains the secret key, then the creator can correctly decrypt the deniable encryptions. If the creator simply deletes the secret key, then the creator cannot decrypt the deniable encryptions; the deniable encryptions will effectively be random data with respect to the creator. This approach fails since there is no way for the creator to prove that the secret key was erased. In contrast, in a questionable encryption scheme the creator can prove that the he/she *never had* and *never will* have the ability to decipher the questionable encryptions.

8 Conclusion

The notion of a *questionable encryption* was defined and an instantiation based on the Paillier public key cryptosystem was given. Questionable encryptions were shown to be related yet distinct from various forms of oblivious transfer. We showed how to use questionable encryptions to build a stronger case in practice that mobile agents that appear to transmit host data using asymmetric cryptography may not in fact do so. This enhances the privacy of the operation of mobile agents.

References

1. E. Bach, J. Shallit. Algorithmic Number Theory - Volume I: Efficient Algorithms. Chapter 7 - Solving Equations over Finite Fields, MIT Press, 1996.
2. M. Blum, S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *Advances in Cryptology—Crypto '84*, pages 289–302, 1985.
3. L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. In *SIAM Journal on Computing*, v. 15, no. 2, pages 364–383, 1986.
4. G. Brassard, C. Crépeau, J. M. Robert. All-or-nothing disclosure of secrets. In *Advances in Cryptology—Crypto '86*, pages 234–238, 1986.
5. G. Brassard, C. Crépeau, J. M. Robert. Information Theoretic Reductions among Disclosure Problems. In *IEEE Symposium on Foundations of Computer Science*, pages 168–173, 1986.
6. R. Berger, R. Peralta, T. Tedrick. A Provably Secure Oblivious Transfer Protocol. In *Advances in Cryptology—Eurocrypt '84*, pages 379–386, 1985.
7. M. Blum. Three applications of the oblivious transfer: Part I: Coin flipping by telephone; Part II: How to exchange secrets; Part III: How to send certified electronic mail. UC Berkeley, 1981.
8. D. Boneh. The Decision Diffie-Hellman Problem. In proceedings of the Third Algorithmic Number Theory Symposium, pages 48-63, 1998.
9. C. Cachin, S. Micali, M. Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In *Advances in Cryptology—Eurocrypt '99*, pages 402-414, 1999.
10. R. Canetti, C. Dwork, M. Naor, R. Ostrovsky. Deniable Encryption. In *Advances in Cryptology—Crypto '97*, pages 90–104, 1997.
11. R. Cramer, V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—Crypto '98*, pages 13–25, 1998.
12. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In IEEE Trans. Inform. Theory. v. 31, pages 469–472, 1985.
13. S. Goldwasser, M. Bellare. Lecture Notes on Cryptography. Manuscript, July 10, 1996.
14. S. Goldwasser, S. Micali. Probabilistic Encryption. JCSS, v. 28, n. 2, pages 270–299, 1984.
15. J. Kilian. Founding cryptography on oblivious transfer. In ACM STOC, pages 20–31, 1988.
16. J. Kilian. Uses of randomness in algorithms and protocols. MIT Press, 1990.
17. E. Kushilevitz, R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In IEEE Foundations of Computer Science, pages 364–373, 1997.
18. PKCS #1-RSA Cryptography Standard, version 2.1, available from www.rsa.com/rsalabs/pkcs.
19. P. Paillier. Public-Key Cryptosystems based on Composite Degree Residue Classes. In *Advances in Cryptology—Eurocrypt '99*, pages 223–238, 1999.

20. M. Rabin. How to exchange secrets by oblivious transfer. Harvard Aiken Comp. Lab, TR-81, 1981.
21. R. Rivest, A. Shamir, L. Adleman. A method for obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the ACM*, v. 21, n. 2, pages 120–126, 1978.
22. Y. Tsiounis, M. Yung. On the security of ElGamal-based encryption. International Workshop on Practice and Theory in Public Key Cryptography, pages 117-134, 1998.
23. A. Young, M. Yung. Deniable Password Snatching: On the Possibility of Evasive Electronic Espionage. IEEE Symposium on Security and Privacy, pages 224–235, 1997.

A Related Primitives for Privacy

A (1,2)-oblivious transfer is a way for Alice to give Bob one out of two possible messages such that Alice has no way of knowing which of the two was received [7,20,6,15,16]. Bob has no control over which of the messages he receives, but knows that he will receive one. A questionable encryption scheme implements a form of oblivious transfer in which the recipient has complete control over whether the message is received. Such variants are called *all-or-nothing disclosure* [4,5]. This is the same as in a questionable encryption, so a questionable encryption scheme can be viewed as an all-or-nothing disclosure variant.

However, a critical difference between these two notions is as follows. A questionable encryption scheme is a "cipher" that can be applied repeatedly and independently to many pieces of data, not data defined within a scope of a single protocol as in all-or-nothing disclosure. In a questionable encryption scheme, the sender need only obtain the "public key" of the receiver once, and from then on messages (i.e., "encryptions") are sent in a one-way fashion from the sender to the receiver in an all-or-nothing disclosure. A questionable encryption scheme allows Bob to prove whether everything was received or nothing was received by revealing the witness.

Our use of witnesses is different but related to the use of witnesses in *deniable encryptions* [10]. In our scheme, Bob has a witness that the value is an encryption or non-encryption under a particular asymmetric encryption function. In a *deniable encryption*, Bob can present a witness for each possible interpretation of the plaintext. Another difference lies in the operational setting. In a deniable encryption scheme, Alice and Bob share a secret key that allows Bob to identify the correct plaintext among the possible plaintexts and deniability is a stronger requirement that allows the receiver to claim any message to an observer (thus it requires specialized implementations and less efficient ones, and it is essentially a symmetric key encryption due to the shared key). In our scheme Alice only knows the 'public key' of Bob and her own secret plaintexts.

B Efficient Implementation

We implemented our questionable encryption scheme in Cygwin using the OpenSSL-0.9.7e-1 cryptographic library. Our improvements are based on Section 7 of [19]. However, we digress from Section 7 by using a more efficient version of the Chinese Remaindering algorithm. We utilize the fast Chinese Remaindering method in RSA's PKCS #1 standard [18] (i.e., the use of $gInv$). Our computation of m during Paillier decryption therefore utilizes zero calls to the Extended Euclidean Algorithm instead of two calls.

Fast Encryption: The public key data structure consists of (n, g, n^2). This avoids having to square n during encryption. It is possible to store the array $(g, g^2, g^4, ...)$ as part of the public key. This speeds up the computation of $g^m \bmod n^2$. However, we opted not to do this type of optimization.

Fast Decryption: The private key data structure consists of the tuple,

$$(p, q, p^2, q^2, \text{pinvmod2tow}, \text{qinvmod2tow}, h_p, h_q, qInv)$$

We have that,

$$\text{pinvmod2tow} = p^{-1} \bmod 2^{|p|}$$
$$\text{qinvmod2tow} = q^{-1} \bmod 2^{|q|}$$
$$h_p = L(g^{p-1} \bmod p^2, p)^{-1} \bmod p$$
$$h_q = L(g^{q-1} \bmod q^2, q)^{-1} \bmod q$$

Adopting the variable name from PKCS #1, we have that $qInv = q^{-1} \bmod p$. Instead of using L as previously defined, we use the speedup mentioned by Paillier that performs operations modulo a power of 2. More specifically, we use LFast (u,n,ninv) $= (u-1)*\text{ninv} \bmod 2^{|n|}$. The efficient Paillier decryption algorithm is given below.

1. compute $m_p = \text{LFast}(c^{p-1} \bmod p^2, p, \text{pinvmod2tow})\ h_p \bmod p$
2. compute $m_q = \text{LFast}(c^{q-1} \bmod q^2, q, \text{qinvmod2tow})\ h_q \bmod q$
3. compute $h = (m_p - m_q)qInv \bmod p$
4. output $m = m_q + qh$ and halt

Fast G_1: To generate g in G_1 we generate $g_p \in \mathbb{Z}^*_{p^2}$ to be a random element with order divisible by p and $g_q \in \mathbb{Z}^*_{q^2}$ to be a random element with order divisible by q. These are then Chinese Remaindered to compute the public key parameter g. We use LFast to compute h_p and h_q.

Fast G_0: Let $p = 2p_1 + 1$ and $q = 2q_1 + 1$. In an analagous improvement, to generate g in G_0 we generate $g_p \in \mathbb{Z}^*_{p^2}$ to be a random element having order $2p_1$ or p_1. We generate $g_q \in \mathbb{Z}^*_{q^2}$ to be a random element having order $2q_1$ or q_1. It is required that at least one of these orders be even. g_p and g_q are then Chinese Remaindered to compute the public key parameter g. We use LFast to compute h_p and h_q.

Fast F: The function F can be implemented efficiently by analyzing g efficiently. This can be accomplished by computing $g_p = g \bmod p^2$ and $g_q = g \bmod q^2$ and then determining the order of g_p and g_q. This reveals the order of g. Note that the order of g_p can be found at about the cost of computing three exponentiations modulo p^2 in which the exponent is approximately $|p|$ bits in length.

Twin RSA

Arjen K. Lenstra[1,2] and Benjamin M.M. de Weger[2]

[1] Lucent Technologies, Bell Laboratories, Room 2T-504,
600 Mountain Avenue, P.O.Box 636, Murray Hill, NJ 07974-0636, USA
[2] Technische Universiteit Eindhoven,
P.O.Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. We introduce *Twin RSA*, pairs of RSA moduli $(n, n+2)$, and formulate several questions related to it. Our main questions are: is Twin RSA secure, and what is it good for?

Keywords: recreational cryptography.

1 Introduction

Regular RSA moduli are constructed by multiplying two more or less randomly selected primes of appropriate sizes. As a result, representation of a regular $2N$-bit RSA modulus requires about $2N$ bits. To save on the representation size of RSA moduli, several methods were proposed in [8], some of which were broken in [1]. An often reinvented folklore approach to generate $2N$-bit RSA moduli that can be represented using just N bits, published in [5] along with several simple variants, still seems to be unbroken. This simple method works as follows. For an N-bit number x that is known from the context, repeatedly select an N-bit prime p at random until the integer part q of the quotient $(x+1)2^N/p$ is prime, then the most significant N bits of the RSA modulus $n = pq$ are given by x. Faster variants add some slack to x and replace q by $q+1$ until it is prime, but the principle remains the same. Since x is known from the context—or can for instance be chosen as 2^{N-1}—the N least significant bits suffice to represent the $2N$-bit RSA modulus n. If one is willing to also consider moduli of unbalanced factor sizes, e.g. a product of primes of sizes $\frac{1}{2}N$ and $\frac{3}{2}N$, respectively, then, as was shown in [5], a $2N$-bit modulus can even be represented using $\frac{1}{2}N$ bits. In particular this shows that pairs of RSA moduli can be generated in such a way that the pair can be represented using the space of a single regular or even a half unbalanced RSA modulus.

In this note we present a method that achieves the same 'compression ratio' for pairs of RSA moduli, in a different and esthetically more pleasing way. Our method is implicit in one of the methods described in [6] and thus not new. The reason we present this particular case of the method from [6] is the fact that the possibility of the construction is usually met first with amazement, quickly followed by skepticism about the security, and finally with puzzled resignation that the resulting moduli indeed look hard to break. Thus, we would like to offer it as a challenge to a wider audience, hoping for either a better security argument than what can be found in [6], or a more effective cryptanalysis.

Another question we want to pose with this note is: are there any applications of Twin RSA that are more interesting than the ones we have been able to offer so far? We realize that it is by no means good marketing policy to present a new cryptographic method without convincing evidence of its practical potential or cryptographic significance. On the other hand, publishing the method despite the fact that we cannot think of a sensible application ourselves, at least has the potential to uncover new possibilities by bringing it to the attention of members of the practical cryptographic community who may never have realized that such remarkable pairs of RSA moduli were possible—or secure.

The remainder of this note is organized as follows. Our method to generate RSA moduli with a fixed prescribed difference is described and discussed in Section 2. A few generalizations are offered in Section 3, and Section 4 concludes this note with two factoring challenges.

2 Twin RSA

Generation of RSA moduli with any prescribed even integer difference d is an easy application of the Chinese Remainder Theorem. The details are described in Algorithm 1 below.

Algorithm 1. Let $d \neq 0$ be a small fixed even integer and let $2N$ be the bitlength of the RSA moduli to be generated.

1. Select two random N-bit primes p and q.
2. Use the Chinese Remainder Theorem to calculate the least positive integer n such that $n \equiv 0 \bmod p$ and $n \equiv -d \bmod q$ and let $r = n/p$ and $s = (n+d)/q$.
3. If n or $n + d$ does not have bitlength $2N$, or if r or s is composite, then return to Step 1.
4. Output the pair of RSA moduli $(n, n + d)$ with factorizations $n = pr$ and $n + d = qs$.

For actual RSA applications of the resulting moduli, co-primality requirements with respect to one's favorite public exponent(s) and $p-1$, $q-1$, $r-1$, and $s-1$ have to be included in the above description.

Twin RSA. We introduce the term *Twin RSA* for the pair of moduli that results from Algorithm 1 when $d = \pm 2$.

Abundance. A single moment of reflection learns that it is most likely the case that Twin RSA moduli are abundant. The Prime Number Theorem combined with the assumption that the factorizations of n and $n+2$ are independent leads to the conjecture that the number of Twin RSA moduli up to x is asymptotically equal to $cx/(\log x)^4$, for some positive constant c. The same argument applies to the general case $(n, n + d)$ for odd d.

Runtime of Algorithm 1. Based on the Prime Number Theorem and the runtime of a single probabilistic compositeness test (using standard arithmetic), one may expect that each execution of Step 1 of Algorithm 1 takes expected runtime $O(N \times N^3 + N \times N^3) = O(N^4)$. Assuming that r and s behave as independent random N-bit numbers, they will simultaneously be prime with probability proportional to $1/N^2$, again based on the Prime Number Theorem. Using standard arithmetic, the computation in Step 3 of Algorithm 1 can be expected to take runtime $O(N^3 + (1/N) \times N^3) = O(N^3)$ (where the factor $1/N$ accounts for the probability that r is not found to be composite, in which case compositeness of s has to be tested as well) and dominates the runtime of the computation in Step 2. Overall, we find that the expected runtime becomes $O(N^2(N^4 + N^3)) = O(N^6)$.

Practical considerations. A practical speed-up can be obtained by generating the sequence of candidate p's in Step 1 of Algorithm 1 using sieving based methods, independently from the similarly generated sequence of candidate q's. Also, upon return to Step 1, one may decide to generate a new p or a new q, but not both.

A more substantial speed-up is obtained by allowing more candidate quotients per prime pair (p, q): in Step 3 of Algorithm 1, add some slack to the lengths and find the smallest positive integer k such that the quotients $(n+kpq)/p$ and $(n + kpq + d)/q$ are both prime. The resulting method can be made to work in expected time $O(N^5)$, but results in RSA moduli pairs that are somewhat less 'elegant' because their factors will have slightly different sizes. The more time one is willing to invest in the generation of RSA moduli pairs with fixed prescribed difference, the closer factor sizes one will be able to obtain.

Independent generation of n and $n + d$? Given d, the moduli n and $n + d$ are generated simultaneously by the single party that executes Algorithm 1. As a result, that same party knows the factorizations of both moduli. We are not aware of a simple variant of our approach where a first party generates p, a second party generates q, and the two parties engage in a straightforward protocol that results in RSA modulus n for the first party and $n + 2$ for the second party, without either party knowing the factorization of the other party's modulus. We pose the challenge of developing such a protocol because it may lead to more interesting applications of Twin RSA.

Security? With variable d, and as argued in [6], it seems impossible to distinguish an RSA moduli pair $(n, n + d)$ generated using our method from a pair of regular RSA moduli that happens to have difference d. This suggests that 'our' pairs $(n, n + d)$ are as secure as 'regular' pairs: being able to do the private RSA operation for either modulus is independent of whether the private RSA operation can be carried out for the other modulus or not.

But what about a fixed choice of d, such as $d = 2$? Most certainly, and intuitively, Twin RSA looks highly suspicious. A more subtle security argument is required in this case, which is one of the challenges we pose with this note. All we can present at this point in support of our belief in the security of Twin RSA—if one is willing to believe in the hardness of factoring to begin with—is the circumstantial evidence that generations of factorers who contributed to the Cunningham factoring project (cf. [3]), where factorizations of $b^k \pm 1$ are collected for $2 \leq b \leq 12$ up to high powers k, have never been able to profit from the factorization of a certain $b^k \pm 1$ to factor the corresponding $b^k \mp 1$.

We also believe that, even when the two moduli share the same public exponent e (such as in practice often happens, e.g. $e = 65537$), the two corresponding private exponents will be completely different and independent for all practical purposes. So, here we do not see any reason for additional suspicion either.

Applications? As mentioned in the Introduction, one of our reasons to publish this note is that we are curious to know if there are any interesting applications of Twin RSA. Here the 'application' may either wear a white or a black hat, as long as it enables us to do something we were unable to do before. Some rather unconvincing white hat applications use the twin modulus as backup in case the other one is believed to be compromised, or use one for encryption and the other for signature purposes—conveniently avoiding the cost of two different certificates for the two different keys. In situations where the size of two standard X.509 certificates is too costly, e.g. because of memory or bandwidth restrictions such as in the mobile telephony world, Twin RSA can be useful. We can envisage one certificate, in which one Certificate Authority (CA) signature is used to bind the owner's identity information to two RSA key pairs (a Twin RSA pair), which is represented by just one of the two moduli and a common (usually very small) public exponent. This will save almost half of the space needed to represent the public key (and when predetermined bits are used, as explained in the next section, a reduction to 25% even becomes possible). One of the conditions that has to be posed is that there must be a standardized way of interpreting this public key representation, of how to extract both public keys, and of establishing the allowed key usages for both keys. It should be possible to do this with only minor adaptations to existing certificate standards such as X.509. Another condition is that the CA should explicitly guarantee that it has seen proof that the certificate owner is in possession of both private keys.

Are there applications with more cryptographic significance? And are there any applications with 'interesting' cryptanalytic potential?

3 Generalizations

Multiple RSA. With a proper choice of even differences d_i for $0 \leq i < t$ and $d_0 = 0$, our method allows construction of t-tuples of RSA moduli $(n + d_i)_{i=0}^{t-1}$, if one is willing to accept moduli where the size of the largest of the two factors is $t - 1$ times the size of the smallest one. For large modulus sizes this may be acceptable, and possibly even desirable (cf. [7]). Note that, for instance, $d_1 = 2$, $d_2 = 4$ will not work: the set $\{d_i \bmod 3 : i = 0, 1, 2\}$ equals the full residue set $\{0, 1, 2\}$ modulo the prime 3, so that for any integer n there will always be an $i \in \{0, 1, 2\}$ such that $n + d_i$ is divisible by 3. More in general, the set consisting of the d_i's, for $0 \leq i < t$, should not contain a full residue system modulo any prime $\leq t$. The latter condition is obviously necessary, but also sufficient. This can easily be seen as follows: for each prime $q \leq t$ there exists an

$a_q \in \{0, 1, \ldots, q-1\}$ such that $d_j \not\equiv a_q \bmod q$ for all j. We now restrict ourselves to n that are equal to $-a_q \bmod q$ for all $q \leq t$, which can be obtained by Chinese Remaindering. Then q does not divide $n + d_j$ for all $q \leq t$ and all j.

Algorithm 2. Let $t \geq 2$, let $(d_i)_{i=0}^{t-1}$ with $d_0 = 0$ be a set of small even integers that does not contain a full residue system modulo any prime $\leq t$, and let tN be the bitlength of the RSA moduli to be generated.

1. Select t random N-bit primes p_i, $0 \leq i < t$.
2. Use the Chinese Remainder Theorem to calculate the least positive integer n of bitlength at most tN such that $n \equiv -d_i \bmod p_i$ for $0 \leq i < t$, and let $n_i = n + d_i$ and $r_i = n_i/p_i$ for $0 \leq i < t$.
3. If n_i does not have bitlength tN for an $i \in \{0, 1, \ldots, t-1\}$ or if there is an $i \in \{0, 1, \ldots, t-1\}$ such that r_i is composite, then return to Step 1.
4. Output the t-tuple of RSA moduli $(n_i)_{i=0}^{t-1}$ with factorizations $n_i = p_i r_i$.

As before, Algorithm 2 can be seen to have expected runtime $O(N^{t+4})$.

Twin RSA with predetermined bits. Our methods can simply be combined with the methods from [5], again at the cost of severely unbalancing the factor sizes (cf. [6]), in a way similar to the construction of the colliding X.509 certificates that were reported in [2]. For instance, for any N-bit number x and assuming that N is even, a pair of $2N$-bit RSA moduli $(n, n+d)$ can be constructed such that the leading N bits of n are given by x, and such that both n and $n+d$ are the products of a $3N/2$-bit prime and an $N/2$-bit prime. In the interest of space we elaborate.

Algorithm 3. Let $d \neq 0$ be a small fixed even integer, let x be an integer with $2^{N-1} \leq x < 2^N$ for some even positive integer N such that $2N$ is the bitlength of the RSA moduli to be generated.

1. Select two random $N/2$-bit primes p and q.
2. Use the Chinese Remainder Theorem to calculate the least positive integer $b < pq$ such that $b \equiv -x2^N \bmod p$ and $b \equiv -x2^N - d \bmod q$, let $n = x2^N + b$, and let $r = n/p$ and $s = (n+d)/q$.
3. If r or s is composite, then return to Step 1.
4. Output the pair of RSA moduli $(n, n+d)$ with factorizations $n = pr$ and $n + d = qs$. The most significant N bits of n and $n+d$ are the same and are given by x.

Remarks. None of the approaches mentioned in this or the previous section yields a representation saving that is larger than what can be obtained by using just the methods from [5].

The practical speed-up tricks that applied to Algorithm 1 can, with the obvious changes, be applied to Algorithms 2 and 3 as well. Furthermore, Algorithm 3 allows a similar generalization to t-tuples—Multiple RSA with predetermined bits—as was presented for Algorithm 1 in Algorithm 2. The restriction to even N in Algorithm 3 is not crucial and just for ease of exposition. Different sized predetermined parts can be handled in essentially the same way. Putting the predermined part in the least significant bits, or spreading it over most and significant bits, is possible too. The underlying idea of all these generalization is the same: combine any of the methods described in [5] with the Chinese Remainder Theorem.

We gladly leave other variants of our idea, namely *Twin Discrete Log*, or subvariants such as *Twin (Hyper)Elliptic Discrete Log*, for others to pursue. This could take the not so challenging form of twin prime fields $(\mathbf{F}_p, \mathbf{F}_{p+2})$, of equally uninteresting generators $(g, g+1)$ (no need to jump by 2 this time!) of the same full multiplicative group, or, the sole interesting case (cf. [6]), of generators $g, g+1$ of a relative small prime order subgroup of a multiplicative group of a prime field—actually, the range of possibilities is only limited by one's imagination. What it would be good for is an entirely different matter.

4 Factoring Challenges

Below the n is given of a Twin RSA pair $(n, n+2)$ consisting of two 1024-bit RSA moduli that each have one 256-bit prime factor, along with the 256-bit factor p of n. Note that the Twin RSA pair $(n, n+2)$ can be represented using just 512 bits. Finding the 256-bit factor of $n+2$ should be borderline possible using the elliptic curve factoring method (ECM), now that a 219-bit (66-digit) factor found using ECM has been announced [4]. Can anyone factor $n+2$ faster, if at all possible using the known factorization of n?

```
n = 80000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    11B9E917 7E937E9D 6AAB2AB0 28940F89 BEEC962C 286F28A9 DB965A18 688EE789
    ACF43457 0E44D41B F837B4EF 9E435CFB 56C2E2F7 00EE8DDD 2A3ECF9F B2EA360D

p = EF0650E4 304D1242 F3DBAF45 80BFB645 77527C60 3C1E3006 BCDE98FB 5F97E507.
```

Obviously, with the fixed prefix this size cannot be recommended for practical purposes. A more substantial factoring challenge is given by the Twin RSA pair $(m, m+2)$, for which the 2048-bit m is given below along with its 512-bit factor q. Finding the 512-bit prime factor of $m+2$ using ECM can be expected to be about as hard as factoring $m+2$ using the Number Field Sieve. Can $m+2$ be factored in an easier way, possibly using the known factorization of m?

```
m = 80000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    396099A3 5F9D2B49 E7BB729E 9542A7B0 A1FAD34B EE884199 E29A5DB4 E49DE1C8
    279682F4 2A92FBFF 4F0F891F 65638997 B28D26DA 10B7529A 40CFA534 8BB95BE8
    ADF4A21B 7DC562D4 93590D53 6B6124C5 6DB5D693 1004A7B4 C031C401 A4B6E1E8
    EA5C8362 E7B2DB3F BFDEF87D 75311FEA 7D9BF1C3 9E3E64DF 9163E468 6D5D2711

q = C1A25EAE FF7A187A 6F793972 199F192B 3D2912DF BD4586CF 4D1CE614 6D75992F
    AB3A7E2A 2149CD3C 4FE9F8A4 8DF3B515 74660F13 696BC4BD 4808E475 69E414B9.
```

As far as we know this last Twin RSA size can be recommended for practical purposes, either with or without a fixed 1024-bit prefix. With the prefix it allows representation of two 2048-bit moduli at the cost of representing 1024 bits.

References

1. D. Coppersmith, *Finding a small root of a bivariate integer equation; factoring with high bits known*, Eurocrypt'96, LNCS 1070, Springer-Verlag 1996, 178–189.
2. Colliding X.509 Certificates, see `http://www.win.tue.nl/~bdeweger/CollidingCertificates/`.
3. Cunningham project, see `http://www.cerias.purdue.edu/homes/ssw/cun/`.
4. B. Dodson, *email announcement of the ECM factorization of M963*, April 6, 2005.
5. A.K. Lenstra, *Generating RSA moduli with a predetermined portion*, Asiacrypt'98, LNCS 1514, Springer-Verlag 1998, 1–10.
6. A.K. Lenstra, B.M.M. de Weger, *On the possibility of constructing meaningful hash collisions for public keys*, ACISP 2005, LNCS 3574, Springer-Verlag 2005, 267–279.
7. A. Shamir, *RSA for paranoids*, RSA Laboratories' Cryptobytes, v. 1, no. 3 (1995) 1–4.
8. S.A. Vanstone, R.J. Zuccherato, *Short RSA keys and their generation*, J. Cryptology, **8** (1995) 101–114.

Security of Two-Party Identity-Based Key Agreement

Colin Boyd and Kim-Kwang Raymond Choo

Information Security Institute,
Queensland University of Technology,
GPO Box 2434, Brisbane Q4001, Australia
boyd@isrc.qut.edu.au, k.choo@qut.edu.au

Abstract. Identity-based cryptography has become extremely fashionable in the last few years. As a consequence many proposals for identity-based key establishment have emerged, the majority in the two party case. We survey the currently proposed protocols of this type, examining their security and efficiency. Problems with some published protocols are noted.

1 Introduction

One of the main purposes of using public-key cryptography, in comparison to shared-key cryptography, is to make key distribution easier. Public keys by their nature need not be kept confidential. On the other hand, integrity of public keys is critical for security and therefore public key certificates have been used for many years. Management of public key certificates has proven to be a harder task than was initially realised and so new directions have been sought. Identity-based cryptography removes the need for certificates since the identity of the owner is the public key. Such public keys can include any descriptive information including temporal information.

Public key cryptography (and identity-based cryptography in particular) only addresses management of long-term public keys which are not suitable for bulk cryptographic processing. For such purposes symmetric keys are usually required which are established freshly for each individual session. Protocols for establishing such *session keys* come in many different types and have a reputation for being difficult to design correctly. One of the simplest and most common types of key establishment protocols are key agreement protocols in which the session key is defined by inputs from the protocol participants.

In the past few years there has been extreme interest in the use of identity-based cryptography, mainly due to the use of elliptic curve pairings to realise cryptographic structures that did not seem possible before. Amongst the many resulting new tools that have been proposed have been a large number of key agreement protocols based on pairings. In the rush to exploit the new ideas many of these protocols have been published without a careful security analysis or a systematic comparison with alternatives. The situation is somewhat like that 20 years ago when key establishment protocols for conventional public key cryptography were routinely published without a proper security analysis.

The purpose of this paper is to make a critical appraisal of the current status of identity-based key agreement protocols, limited to the two-party case. We examine the security properties and efficiency achieved in a large number of published protocols. We emphasise the importance of precise security models and note deficiencies in several protocols.

The rest of this paper is structured as follows. The following section defines the subject matter in more detail by discussing relevant background on identity-based cryptography and key agreement protocols. Section 3 surveys the field of existing published protocols and analyses their comparitive security and efficiency. The conclusion speculates where subsequent progress may be likely.

2 Identity-Based Cryptography and Key Agreement

The original idea for identity-based cryptography goes back to Shamir [30] over 20 years ago. Identity-based cryptography does away with public keys altogether so no certificates are required (although the authenticity of public parameters needs to be assured). This is of great benefit in simplifying key management. However, a drawback of all true identity-based schemes is that users cannot be allowed to generate their own private keys (otherwise anyone could find any user's private key) and therefore key escrow is inevitable.

Shamir gave an algorithm for identity-based signatures but was unable to obtain an identity-based encryption algorithm. However, in 1987 Okamoto [24, 25] published the first identity-based key agreement protocol. It uses a composite modulus n whose factorisation is known only to a trusted authority. The authority chooses values e and d as in the RSA algorithm, so that $ed \bmod \phi(n) = 1$, and an element g that is primitive in both the integers mod p and the integers mod q. The values g and e are made public.

Before engaging in the key agreement protocol each user must register with the authority to obtain a private key. Party P_i's identification string, ID_i, is treated as an integer modulo n. The authority calculates the value $s_i = ID_i^{-d} \bmod n$ and distributes s_i securely to user I. Once this registration is completed users may agree fresh session keys without recourse to any other information other than the fixed parameters e and n and the identity of the partner with which the key is to be shared.

Protocol 1 shows the key agreement message flows. The shared secret is defined as $Z_{AB} = g^{er_Ar_B}$. On the assumption that it is necessary to know either s_A or s_B in order to find Z_{AB}, the scheme prevents an adversary from learning the session key.

Mambo and Shizuya [22] and later Kim *et al.* [18] provided a security proof against active attacks. They showed a reduction of attacks on the protocol to the Diffie–Hellman problem or to the RSA problem. Their model is similar to the Bellare–Rogaway security model [3, 4] discussed below.

$$\begin{array}{ccc}
A & & B \\
r_A \in_R \mathbb{Z}_n & & \\
t_A = g^{r_A} & \xrightarrow{s_A t_A} & r_B \in_R \mathbb{Z}_n \\
 & \xleftarrow{s_B t_B} & t_B = g^{r_B} \\
Z_{AB} = ((s_B t_B)^e ID_B)^{r_A} & & Z_{AB} = ((s_A t_A)^e ID_A)^{r_B}
\end{array}$$

Protocol 1: Okamoto's identity-based protocol

Interest in identity-based cryptography was resurrected when Boneh and Franklin [6] presented the first identity-based encryption scheme using the idea of a bilinear map based on elliptic curve pairings. However, even before this the applications of pairings to identity-based key agreement were recognised by Sakai *et al.* [29]. Before looking at the SOK protocol we have to introduce some notation and concepts about pairings and bilinear maps. Except where noted otherwise, the following notation is used for all protocols in this paper.

Using the notation of Boneh and Franklin [6], we let $\mathbb{G}_1$ be an additive group of prime order q and $\mathbb{G}_2$ be a multiplicative group of the same order q. We assume the existence of a map $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$. Typically, $\mathbb{G}_1$ will be a subgroup of the group of points on an elliptic curve over a finite field, $\mathbb{G}_2$ will be a subgroup of the multiplicative group of a related finite field and the map $\hat{e}$ will be derived from either the Weil or Tate pairing on the elliptic curve. The mapping $\hat{e}$ must be efficiently computable and has the following properties.

Bilinear: for $Q, W, Z \in \mathbb{G}_1$, both

$$\hat{e}(Q, W+Z) = \hat{e}(Q,W) \cdot \hat{e}(Q,Z) \quad \text{and} \quad \hat{e}(Q+W, Z) = \hat{e}(Q,Z) \cdot \hat{e}(W,Z).$$

Non-degenerate: for some element $P \in \mathbb{G}_1$, we have $\hat{e}(P,P) \neq 1_{\mathbb{G}_2}$.

When $a \in \mathbb{Z}_q$ and $Q \in \mathbb{G}_1$, we write aQ for scalar multiplication of Q by a. Due to bilinearity, for any $Q, W \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$ we have:

$$\hat{e}(aQ, bW) = \hat{e}(Q,W)^{ab} = \hat{e}(abQ, W).$$

Recent literature [1, 2, 6, 15] provides a more comprehensive description of how these groups, pairings and other parameters should be selected in practice for efficiency and security.

A random value $s \in \mathbb{Z}_q$ plays the role of the *master secret* of the *Key Generation Centre* (KGC) in the ID-based system. The KGC distributes to each party P_i with identity ID_i a long-term key pair consisting of public key $Q_i = H_1(ID_i)$ and private key $S_i = sQ_i$. Here H_1 is a hash function mapping identities $ID_i \in \{0,1\}^*$ onto $\mathbb{G}_1$. The KGC also publishes the system parameters which include descriptions of the two groups $\mathbb{G}_1$ and G_2, a point P that generates $\mathbb{G}_1$, and a master public key sP.

SOK Protocol [29]. With the above parameters, any two principals P_i, P_j with identities ID_i, ID_j can efficiently calculate a shared key:

$$F_{ij} = \hat{e}(Q_i, Q_j)^s = \hat{e}(S_i, Q_j) = \hat{e}(S_j, Q_i).$$

This protocol for *identity-based, non-interactive key distribution* is analogous to static Diffie–Hellman but does not require certificates. Dupont and Enge [14] analysed the security of the protocol. Like many identity-based protocols, the security of SOK relies on the difficulty of the *Bilinear Diffie-Hellman Problem (BDHP).* Given $\mathbb{G}_1$, $\mathbb{G}_2$ and $\hat{e}$ as above, the BDHP is to compute $\hat{e}(P, P)^{xyz} \in \mathbb{G}_2$ given $\langle P, xP, yP, zP \rangle$ with $P \in \mathbb{G}_1$ and $x, y, z \in \mathbb{Z}_q$.

At this point it is reasonable to ask what advantage there is in identity-based key agreement based on pairings in comparison with older identity-based protocols such as Okamoto's (Protocol 1 above). Generally the answer may be expected to be the same advantages as using elliptic curves over older public key technology, namely a saving in computation and key size. This is certainly true with regard to savings in bandwidth since message exchanges can be considerably shorter. However, it may not necessarily be the case in terms of computation because the pairing operation can be quite costly. Research is still quite active in deciding how to implement pairings most efficiently. In Section 3.2 we compare the efficiency of many pairings-based key agreement protocols. Another reason for choosing pairings-based key agreement is to exploit the infrastructure for identity-based cryptography with its many other benefits. In the remainder of this paper we look only at pairings-based key agreement.

2.1 Security Properties for Key Agreement

There are many properties that are required for security of any key agreement protocol. These have been discussed by many authors and we refer to the paper of Blake-Wilson and Menezes [5] for an excellent overview. The most basic property is that a passive adversary eavesdropping on the protocol should be unable to obtain the session key. In a modern context we usually require that, far from obtaining the whole key, the adversary cannot even reliably distinguish between the session key and a randomly chosen string of the expected length. We also generally expect the adversary to be an active one, not only able to see all messages sent, but also able to alter, delete and fabricate messages – in short the adversary is in control of the communications on the network. A number of typical attacks lead to additional security properties as follows.

Known key security. It is often reasonable to assume that the adversary will be able to obtain session keys from any session different from the one under attack. A protocol has known-key security if it is secure under this assumption. This is generally regarded as a standard requirement for key establishment protocols.

Unknown key-share security. Sometimes the adversary may be unable to obtain any useful information about a session key, but can deceive the protocol principals about the identity of the peer entity. This can result in principals giving away information to the wrong party or accepting data as coming from the wrong party. Consequently security against unknown key-share attacks is regarded as a standard requirement.

Forward secrecy. When the long-term key of an entity is compromised the adversary will be able to masquerade as that entity in any future protocol runs. However, the situation will be even worse if the adversary can also use the compromised long-term key to obtain session keys that were accepted before the compromise. Protocols that prevent this are said to provide forward secrecy. Since there is usually a computational cost in providing forward secrecy it is sometimes sacrificed in the interest of efficiency. Forward secrecy for identity-based protocols is similar to conventional public key cryptography. However, there is an additional concern since the master key of the KGC is another secret that could become compromised. When this happens it is clear that the long-term keys of all users will be compromised, but it is possible that a protocol can provide forward secrecy in the usual sense but still give away old session keys if the master key becomes known. We will say that a protocol that retains confidentiality of session keys even when the master key is known provides *KGC forward secrecy.*

Key Compromise Impersonation Resistance. Another problem that may occur when the long-term key of an entity A is compromised is that the adversary may be able to masquerade not only *as* A but also *to* A as another party B. Such a protocol is said to allow key compromise impersonation. Resistance to such attacks is often seen as desirable. Another property that is sometimes desired is *deniability*, which ensures that the protocol does not permit a proof that any particular principal took part. Resistance to key compromise impersonation seems to conflict with deniability [7].

Although the informal security properties just discussed are useful concepts in assessing protocols, the modern view is that a formal analysis is a more reliable way to obtain confidence in the security of a protocol. The computational approach to proofs of protocols for key establishment was established by Bellare and Rogaway [3,4]. Several variants and extensions of the model have been used. Here we outline the basic method. The adversary $\mathcal{A}$ is a probabilistic polynomial time algorithm that controls all the communications that take place between all protocol principals. It does this by interacting with a set of *oracles*, each of which represents an instance of a principal in a specific protocol run. Each principal has an identifier U and oracle Π_U^s represents the actions of principal U in the protocol run indexed by integer s. Interactions with the adversary are called oracle *queries.* We now describe each one informally.

$\mathsf{Send}(U, s, m)$. This query allows the adversary to make the principal U run the protocol normally. The oracle Π_U^s will return to the adversary the same next message that an honest principal U would if sent message m according to the conversation so far.

$\mathsf{Reveal}(U, s)$. This query models known key security. If a session key K_s has previously been accepted by Π_U^s then it is returned to the adversary. An oracle is called *opened* if it has been the object of a Reveal query.

Corrupt(U, K). This query models insider attacks and unknown key share attacks by the adversary. The query returns the oracle's internal state and sets the long-term key of U to be the value K chosen by the adversary. The adversary can then control the behaviour of U with Send queries. A principal is called *corrupted* if it has been the object of a Corrupt query.

Test(U, s). Once the oracle Π_U^s has accepted a session key K_s the adversary can attempt to distinguish it from a random key as the basis of determining security of the protocol. A random bit b is chosen; if $b = 0$ then K_s is returned while if $b = 1$ a random string is returned from the same distribution as session keys. This query is only asked once by the adversary.

The security of the protocol is defined by a game played between the adversary and a collection of user oracles. The adversary will interact with the oracles through the queries defined above. At some stage during the execution a Test query is performed by the adversary. The target oracle for the test query (and any partner it has) must not have been the subject of a Reveal or Corrupt query. Eventually the adversary outputs its guess (a bit) indicating whether the input to the Test query was the real key or not. Success of the adversary $\mathcal{A}$ is measured in terms of its success in getting this guess correct.

Definition 1. *A protocol P is a* secure key establishment protocol *if:*

- *in the presence of a benign adversary partner oracles accept the same key.*
- *no probabilistic polynomial time adversary can win the above game with probability significantly more than $\frac{1}{2}$.*

Security of a protocol is typically proved by finding a reduction to some well known computational problem whose intractability is assumed. The formal definition of security in the computational models captures most of the attacks mentioned above. Some model variants do not consider forward secrecy, while resistance to key compromise impersonation is usually not modelled.

2.2 An Example

In this section we look at a specific protocol due to Ryu, Yoon and Yoo [27]. This should help to understand the typical structure of identity-based key agreement and illustrate some of the important properties. Figure 2 describes the protocol. Parties A and B choose random values a and b and exchange ephemeral public keys T_A and T_B which are used to form the ephemeral Diffie–Hellman key abP in group $\mathbb{G}_1$. They are also assumed to know each other's identity and can therefore both form the long-term shared key $\hat{e}(Q_A, Q_B)^s$ exactly as in the SOK protocol. At the end of the protocol execution, both A and B will compute session keys of the same value:

$$\begin{aligned} K_{AB} &= \mathcal{H}(A, B, K_A, V_A) = \mathcal{H}(A, B, a \cdot T_B, \hat{e}(S_A, Q_B)) \\ &= \mathcal{H}(A, B, abP, \hat{e}(Q_A, Q_B)^s) \\ &= \mathcal{H}(A, B, K_B, V_B) = \mathcal{H}(A, B, b \cdot T_A, \hat{e}(S_B, Q_A)) \end{aligned}$$

$$\begin{array}{ccc}
A & & B \\
a \in_R \mathbb{Z}_q^* & \xrightarrow{T_A = aP} & b \in_R \mathbb{Z}_q^* \\
K_A = a \cdot T_B & \xleftarrow{T_B = bP} & K_B = b \cdot T_A \\
V_A = \hat{e}(S_A, Q_B) & & V_B = \hat{e}(S_B, Q_A)
\end{array}$$

Protocol 2: Ryu–Yoon–Yoo ID-based authenticated key agreement protocol

$$\begin{aligned}
&= \mathcal{H}(A, B, abP, \hat{e}(Q_A, Q_B)^s) \\
&= K_{BA}
\end{aligned}$$

A Key Replicating Attack. We now describe a new attack in which the adversary succeeds in forcing the establishment of a session, S, (other than the Test session or its matching session) that has the same key as the Test session. In this case the adversary can distinguish whether the Test-session key is real or random by asking a Reveal query to the oracle associated with S. Such an attack has been dubbed a *key replicating attack* by Krawczyk [19]. The attack succeeds if the adversary is allowed to ask a Reveal query, as shown in Figure 1. Both A and B have non-matching conversations at the end of the

$$\begin{array}{ccccc}
A & & \mathcal{A} & & B \\
a \in_R \mathbb{Z}_q^* & \xrightarrow{T_A = aP} & \text{Intercept} & & \\
 & & e \in_R \mathbb{Z}_q^* & \xrightarrow{e \cdot T_A} & b \in_R \mathbb{Z}_q^* \\
K_{A'} = a \cdot e \cdot T_B & \xleftarrow{e \cdot T_B} & \text{Intercept} & \xleftarrow{T_B = bP} & K_{B'} = b \cdot e \cdot T_A \\
V_A = \hat{e}(S_A, Q_B) & & & & V_B = \hat{e}(S_B, Q_A) \\
\multicolumn{5}{c}{K_{AB} = \mathcal{H}(A, B, abeP, \hat{e}(Q_A, Q_B)^s) = K_{BA}}
\end{array}$$

Fig. 1. Execution of Protocol 2 in the presence of a malicious adversary

protocol execution, but have accepted the same session key. This session key is $K_{AB} = \mathcal{H}(A, B, abeP, \hat{e}(Q_A, Q_B)^s) = K_{BA}$, depends on e, an input from $\mathcal{A}$. This is a violation of the "key integrity" property [16] which states that any accepted session key should depend only on inputs from the protocol principals. Since both A and B do not have any matching conversations (they are not partners since their protocol views are different), $\mathcal{A}$ is able to trivially expose a fresh session key by revealing either A or B.

Key Compromise Impersonation. In order to demonstrate that the Ryu–Yoon–Yoo protocol does not achieve key compromise impersonation resilience (as claimed), we assume that the adversary, $\mathcal{A}$, has corrupted player A (using a Corrupt query) and has knowledge of the long-term secret key of A, sQ_A.

$\mathcal{A}$ impersonates B and starts a new protocol execution with A. At the end of this protocol execution, $\mathcal{A}$ is able to compute the session key of A as per protocol specification, as shown below:

$$\begin{aligned} K_{AB} &= \mathcal{H}(A, B, K_E, V_A) = \mathcal{H}(A, B, e \cdot T_A, \hat{e}(S_A, Q_B)) \\ &= \mathcal{H}(A, B, aeP, \hat{e}(Q_A, Q_B)^s) \\ &= \mathcal{H}(A, B, a \cdot T_E, \hat{e}(Q_A, Q_B)^s) \end{aligned}$$

3 Comparing Identity-Based Key Agreement Protocols

In this section we survey a large number of protocols that have been published in the recent literature and assess their security and efficiency. Most of the protocols are defined using two message flows, one in each direction between principals A and B. There have been some one-way protocols proposed [26] but we will not look at these in this survey. Many protocols are also defined in a three message version, typically by adding a "handshake" between the parties to provide confidence that they both hold the same key.

We note that there are many similarities between identity-based key agreement and key agreement using standard public key cryptography. Arguably the aim in designing a good ID-based key agreement protocols is to achieve all the properties of the best conventional key agreement protocols but without the need for certified public key, and at the same time trying to maximise efficiency.

3.1 Protocol Definitions

Tables 1 and 2 summarise the definition of each of the protocols. Those in Table 1 use unauthenticated messages, which means that private keys are not used in their construction. In contrast protocols in Table 2 include some direct authentication information, which is checked by the recipient before proceeding. There are three ingredients which essentially define most of these protocols.

Private key. Most protocols use the private key construction used in the first protocol of Sakai *et al.* which we denote Type I. There are to date a few examples of protocols using an alternative key first suggested by Sakai and Kasahara [28] which we denote Type II.

– Type I: $S_U = sQ_I$
– Type II: $S_U = (s + q_U)^{-1}P$

Note that Type I private keys are members of the elliptic curve group $\mathbb{G}_1$ defined by mapping the identity string ID_I of entity I to the value Q_I using a suitable hash function. Boneh and Franklin [6] suggest an explicit function for a particular elliptic curve which costs one exponentiation in the underlying field. This mapping must also be applied to find the public key Q_I. In contrast Type II private keys use a value q_U which is a hash of ID_U whose output is a scalar in $\mathbb{Z}_q$. The corresponding public key for the Type II private key is $(s + q_U)P$ which can be calculated as $sP + q_UP$. Finally there is a variant of Type II which we denote II'. Type II' keys are defined using a different pairing and use two different public generators P and Q for the inputs of the pairing.

Message structure. In order to obtain the best efficiency most protocols send only one message block typically consisting of one elliptic curve point. Some protocols add a second value which can typically be considered as a signature value which is checked by the recipient before the session key is computed.

Session key construction. There are many different ways that the exchanged messages can be combined in order to derive the session key. Each party uses the received message together with its private long-term key and its short-term random input.

Table 1. Summary of unauthenticated two-message ID-based protocols

Protocol	Private key	Message	Session key
Smart [32]	Type I	$T_A = aP$	$\hat{e}(S_A, T_B) \cdot \hat{e}(S_B, T_A)$
CK [9] #1'	Type I	T_A	$\mathcal{H}(\hat{e}(S_A, T_B) \cdot \hat{e}(S_B, T_A) \parallel abP)$
RYY [27]	Type I	T_A	$\mathcal{H}(A \parallel B \parallel \hat{e}(Q_A, Q_B)^s \parallel abP)$
Shim [31]	Type I	T_A	$\mathcal{H}(A \parallel B \parallel \hat{e}(P, P)^{abs} \cdot \hat{e}(Q_A, P)^{bs} \cdot \hat{e}(P, Q_B)^{as} \cdot \hat{e}(Q_A, Q_B)^s)$
CK [9] #2	Type I	$W_A = aQ_A$	$\hat{e}(Q_A, Q_B)^{s(a+b)}$
CK [9] # 2'	Type I	T_A, W_A	$\mathcal{H}(\hat{e}(Q_A, Q_B)^{s(a+b)} \parallel abP)$
Yi [36]	Type I	W_A	$\hat{e}((a + (W_A)_x)Q_A, (b + (W_B)_x)Q_B)^s$
CJL [12] #2	Type I	T_A	$\mathcal{H}(\hat{e}(P, P)^{abs} \parallel Q_A \parallel Q_B)$
Wang [34]	Type I	W_A	$\hat{e}((\psi_B + b)Q_B, \psi_A + a)Q_A)^{sh}$
MB [23] #1	Type II	$R_A = aQ_B$	$\hat{e}(P, P)^{ab}$
MB [23] #2	Type II'	R_A	$\hat{e}(P, Q)^{ab}$
Xie [35] #1	Type II	R_A	$\hat{e}(P, P)^{ab+b+a}$
Xie [35] #2	Type II'	R_A	$\hat{e}(P, Q)^{ab+b+a}$

Protocols in Table 1 are simple enough that it is possible to reconstruct each one from the summary information. In each protocol the message shown is that sent by A. The corresponding message sent by B is symmetrical. In each protocol A computes a random ephemeral private key a which is a scalar in $\mathbb{Z}_q$. In protocols which use a Type I key exchange, messages are either of the form $T_A = aP$, or of the form $W_A = aQ_A$, or both. Protocols with keys of Type II or II' exchange messages of the form $R_A = aQ_B$ where B is the other party. The session key is shown in the table in symmetrical format which does not show directly how it is constructed. $\mathcal{H}$ denotes some secure hash function; $\parallel$ denote concatenation of two messages. In Wang's protocol $\psi_A = \pi(W_A, W_B)$, where $\pi : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_q^*$ is

Table 2. Summary of authenticated two-party, two-message ID-based protocols

Protocol	Private key	Messages	Session key
KRY [17]	Type I	$T_A, \mathcal{H}(T_A)S_A + a \cdot sP$	$\hat{e}(P,P)^{abs}$
CJL [12] #1	Type I	asP, aS_A	$\mathcal{H}(absP \parallel Q_A \parallel Q_B)$
BMP [7]	Type I	aP (authenticated)	$\mathcal{H}(abP)$
CHLS [11]	Type II	See text	$\mathcal{H}(g^a, b, \ldots)$

a special hash function, and h is the co-factor of the elliptic curve defining $\mathbb{G}_1$. In Yi's protocol, $(W_A)_x$ denotes the x-coordinate of point W_A.

Protocols in Table 2 include direct authentication information as a signature of some sort. The first two protocols in this table are symmetrical and use messages as shown. The BMP protocol [7] is the only protocol shown that exists only in a 3-move version. This protocol provides direct authentication of the ephemeral keys aP and bP. The CHLS protocol [11] is specially designed for use by a client of low computational power and consequently its structure is very different from the other protocols listed. Essentially the client sends an encrypted and signed secret value g^a which can be recovered and authenticated by the server. The server sends its input b in cleartext and both parties can then compute the session key as a hash of g^a, b and other values.

There are some interesting comparisons possible between the protocols seen in Table 1 and various protocols using conventional Diffie–Hellman in finite fields. For example, the RYY protocol has strong similarities to the so called Unified Model protocol which is included in the IEEE P1363 standard. There is a close similarity also between the Yi protocol and the MQV protocol. Finally the CK protocol is closely related to MTI A(0) protocol. (Blake-Wilson and Menezes [5] include descriptions of each of these protocols.) These similarities may extend to the security properties of these protocols, though this is currently unproven.

Some protocols include versions that can work with different domains in which separate KGCs use different master keys. These include the CK, MB, and Xie protocols. A protocol of Lee *et al.* [20] (not included in the table) is essentially the same as the CK protocol extended to domains in which different groups are used.

3.2 Protocol Efficiency

Table 3 summarises the computation of each party. We only record multiplications and pairings in group $\mathbb{G}_1$, and exponentiations from $\mathbb{G}_2$. For simplicity we equate exponentiations in $\mathbb{G}_2$ with multiplications in $\mathbb{G}_1$ and add them to the total for M, while the pairings are denoted P.

Computational requirements are divided into two parts, online and offline. The offline computations are those that can be computed before the protocol run starts. We have counted as offline those computations that require knowledge of the identity of the peer. This may not always be realistic. Some computations are

Table 3. Computational requirements for two-party, two-message ID-based protocols

Protocol	**Computation** On-line	**Computation** Off-line
Smart [32]	$1P$	$2M + 1P$
CK [9] #1'	$1M + 1P$	$2M + 1P$
CK [9] #2	$1P$	$2M$
CK [9] #2'	$1M + 1P$	$2M$
Wang [34]	$2M + 1P$	$1M$
Yi [36]	$2M$	$1M + 1P$
RYY [27]	$1M$	$1M + 1P$
KRY [17]	$2M + 3P$	$3M$
CJL [12] #1	$2M + 3P$	$2M$
CJL [12] #2	$1M + 2P$	$1M$
Shim [31]	$1P$	$2M$
Xie [35] #1	$1M + 1P$	$2M + 1P$
Xie [35] #2	$1M + 1P$	$2M + 1P$
MB [23] #1	$1M + 1P$	$1M$
MB [23] #2	$1M + 1P$	$1M$
BMP [7]	$1M$	$2M + 1P$
CHLS [11]	$0/(2P + 2M)$	$4M/0$

also independent of the peer's identity. For the CHLM protocol the computation is different for the client (shown first) and the server (shown second).

The amount of communication bandwidth required in each protocol can be estimated by looking at the messages sent in Tables 1 and 2. Well known techniques for elliptic curve point compression allow points to be expressed as an element in the underlying field plus a single bit. The bandwidth used is considerably less than the RSA-based Protocol 1 if only one point is sent. Protocols that require online pairings computation may be rather inefficient since a pairing requires several times the computation of an elliptic curve multiplication. However, the exact computation required varies considerably depending on the choice of curve and various implementation details. Research is continuing in this area [1].

Most protocol descriptions ignore the cofactor that may be required to ensure that the point sent is a member of the prime order subgroup. Such a check may be important for security reasons (to avoid small subgroup attacks such as those by Lim and Lee [21]). However, when the received point is used in a pairing the effort required to check that the point is in $\mathbb{G}_1$ is only a small part of the overall computation required.

3.3 Protocol Security

We now look at the security of these protocols. Table 4 notes whether each protocol provides forward secrecy, key compromise impersonation resistance (KCIR) and has a security proof. Most proofs have been attempted in the Bellare–

Rogaway (1993) model [3]. However, some of the original proofs have run into trouble and the table shows that many protocols have proofs only in a restricted form in which the adversary is prevented from asking any Reveal queries.

The CHLS and Wang protocols have proofs in the (full) Bellare & Rogaway (1993) model [3] while the BMP protocol has a proof in the Canetti–Krawczyk model [8]. The CK and BMP protocols are proven secure based on the Bilinear Diffie–Hellman (BDH) assumption while the Wang protocol is proven secure using a stronger decisional version of BDH (i.e., DBDH). The security of the Xie and MB protocols assumes the intractability of the Bilinear Inverse Diffie–Hellman (BIDH) problem, which has been proven to be polynomial time equivalent to the BDH problem [37]. The CHLS protocol is based on two assumptions: the modified BIDH with k values (k-mBIDH) and the Collusion Attack Algorithm with k traitors (k-CAA), which are stronger than the BDH assumption.

Table 4. Security properties for two-party, two-message ID-based protocols

Protocol	**Fwd. Secrecy**	**KCIR**	**Security proof**
Smart [32]	No	Yes	No
CK #1' [9]	Yes	Yes	No
CK #2 [9]	No	Yes	Restricted (BDH)
CK #2' [9]	No	Yes	Restricted (BDH)
Wang [34]	No	Yes	Yes (DBDH)
Yi [36]	Yes	Yes	No
RYY [27]	No	No	No (See Sec. 2.2.)
KRY [17]	Yes (No KGC – FS)	Yes	No
CJL [12] #1	Yes	Yes	No (Key replicating attack)
CJL [12] #2	Yes (KGC – FS)	Yes	No (Key replicating attack)
Shim [31]	No	No	Broken by Sun and Hsieh [33]
Xie [35] #1	Yes (No KGC – FS)	Yes	Restricted (BIDH) [10]
Xie [35] #2	Yes	Yes	Restricted (BIDH) [10]
MB [23] #1	Yes (No KGC – FS)	No	Restricted (BIDH) [10], [13]
MB [23] #2	Yes	No	Restricted (BIDH) [10], [13]
BMP [7]	Yes	No	Yes (BDH)
CHLS [11]	No	Yes	Yes (k-mBIDH & k-CAA)

Krawczyk [19] has pointed out that there is a generic attack against forward secrecy on *any* two-party two-flow protocol for which the messages are not explicitly authenticated. In this attack the adversary first masquerades as A, generates the first protocol flow, and records the reply of B. Later, the adversary can corrupt A and compute the old key in the same way as A would have. The existence of such an attack means that *none* of the protocols in Table 1 can provide forward secrecy. We have taken a more relaxed view of this (as have most authors) and assume that key confirmation will follow which prevents this attack. Note, however, that in most cases there is no proof of forward secrecy.

The key replicating attacks noted for CJL protocols 1 and 2 are similar to that on the RYY protocol described in Section 2.2. As in that case, it is possible to fix this problem by adding a session identifier (the concatenation of the exchanged messages) into the definition of the session key [13].

It is clear from Table 4 that there is a significant lack of ID-based protocols with a full security proof. Understanding of the pitfalls and problems has advanced recently and progress in this area can be anticipated soon.

4 Conclusion

Our survey of two-party identity-based key agreement has shown that there are many protocols which have not received adequate scrutiny. Most published protocols do not carry a security proof so that we cannot be sure what their security properties are – our examples show that they may not be as secure as we may like. We urge caution when proposing new protocols, particularly to ensure that a formal security statement is provided with adequate proof, and also that comparison with the many existing protocols is made. Analogies with previously published protocols with well-proven properties may prove useful.

It is still not clear which is the best protocol for a particular application, nor what are the limitations against further improvement. Some of the protocols that look best from the performance and informal analysis are currently lacking a security proof. Another trend to look out for is proofs in the standard model – currently all the proofs that exist rely on random oracles.

In addition to two-party protocols, tripartite and multi-party identity-based key agreement protocols are currently being widely proposed. The correct security model in these cases is even more uncertain but we can expect useful progress in this area in line with the recent advances in security proofs for multi-party key agreement with conventional public key cryptogaphy.

Acknowledgements. This work was supported by the Australian Research Council through Discovery Project DP0345775.

References

1. P. S. L. M. Barreto, S. Galbraith, C. O. hEigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. Cryptology ePrint Archive, Report 2004/375, 2004. `http://eprint.iacr.org/2004/375/`.
2. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - Crypto 2002*, Vol. 2442/2002 of LNCS, pages 354–368. Springer-Verlag, 2002.
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology - Crypto 1993*, pages 110–125. Springer-Verlag, 1993. Vol. 773/1993 of LNCS.
4. M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *27th ACM Symposium on the Theory of Computing - STOC 1995*, pages 57–66. ACM Press, 1995.
5. S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellman Key Agreement Protocols. In *Selected Areas in Cryptography - SAC 1998*, pages 339–361. Springer-Verlag, 1998. Vol. 1556/1998 of LNCS.

6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):585–615, 2003.
7. C. Boyd, W. Mao, and K. Paterson. Key Agreement using Statically Keyed Authenticators. In *Applied Cryptography and Network Security - ACNS 2004*, pages 248–262. Springer-Verlag, 2004. Vol. 3089/2004 of LNCS.
8. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – Eurocrypt 2001*, Vol. 2045/2001 of LNCS, pages 453–474. Springer-Verlag.
9. L. Chen and C. Kudla. Identity Based Authenticated Key Agreement Protocols from Pairings (Corrected version at `http://eprint.iacr.org/2002/184/`). In *16th IEEE Computer Security Foundations Workshop - CSFW 2003*, pages 219–233. IEEE Computer Society Press, 2003.
10. Z. Cheng and L. Chen. On Security Proof of McCullagh-Barreto's Key Agreement Protocol and its Variants. Cryptology ePrint Archive, Report 2005/201, 2005. `http://eprint.iacr.org/2005/201/`.
11. K. Y. Choi, J. Y. Hwang, D. H. Lee, and I. S. Seo. ID-based Authenticated Key Agreement for Low-Power Mobile Devices. In *10th Australasian Conference on Information Security and Privacy - ACISP 2005*, pages 494–505. Springer-Verlag, 2005. Vol. 3574/2005 LNCS.
12. Y. J. Choie, E. Jeong, and E. Lee. Efficient Identity-based Authenticated Key Agreement Protocol from Pairings. *Journal of Applied Mathematics and Computation*, pages 179–188, 2005.
13. K.-K. R. Choo, C. Boyd, and Y. Hitchcock. On Session Key Construction in Provably Secure Protocols (Extended version available from `http://eprint.iacr.org/2005/206`). In *1st International Conference on Cryptology in Malaysia - Mycrypt 2005*. Springer-Verlag, 2005. LNCS.
14. R. Dupont and A. Enge. Practical Non-Interactive Key Distribution Based on Pairings. Cryptology ePrint Archive, Report 2002/136, 2002. `http://eprint.iacr.org/2002/136/`.
15. S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory – ANTS-V*, Vol. 2369/2002 of LNCS, pages 324–337. Springer-Verlag, 2002.
16. P. Janson and G. Tsudik. Secure and Minimal Protocols for Authenticated Key Distribution. *Computer Communications*, pages 645–653, 1995.
17. K.-W. Kim, E.-K. Ryu, and K.-Y. Yoo. ID-Based Authenticated Multiple-Key Agreement Protocol from Pairings. In *International Conference On Computational Science And Its Applications - ICCSA 2004*, pages 672–680. Springer-Verlag, 2004. Vol. 3046/2004 of LNCS.
18. S. Kim, M. Mambo, T. Okamoto, H. Shizuya, M. Tada, and D. Won. On the Security of the Okamoto-Tanaka ID-based Key Exchange Scheme against Active Attacks. *IEICE Transactions Fundamentals*, E84-A(1):231–238, January 2001. `http://search.ieice.or.jp/2001/files/e000a01.htm#e84-a,1,231`.
19. H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol (Extended version available from `http://eprint.iacr.org/2005/176/`). In *Advances in Cryptology - Crypto 2005*. Springer-Verlag, 2005. LNCS.
20. H. Lee, D. Kim, S. Kim, and H. Oh. Identity-based Key Agreement Protocols in a Multiple PKG Environment. In *International Conference On Computational Science And Its Applications - ICCSA 2005*, pages 877–886. Springer-Verlag, 2005. Vol. 3483/2005 of LNCS.

21. C. H. Lim and P. J. Lee. A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup. In *Advances in Cryptology – Crypto 1997*, pages 249–263. Springer-Verlag, 1997. Vol. 1294 of LNCS.
22. M. Mambo and H. Shizuya. A Note on the Complexity of Breaking Okamoto-Tanaka ID-based Key Exchange Scheme. *IEICE Transactions Fundamentals*, E82-A(1):77–80, January 1999.
23. N. McCullagh and P. S. L. M. Barreto. A New Two-Party Identity-Based Authenticated Key Agreement (Extended version available from `http://eprint.iacr.org/2004/122/`). In *Cryptographers' Track at RSA Conference - CT-RSA 2005*, pages 262–274. Springer-Verlag, 2005. Vol. 3376/2005 of LNCS.
24. E. Okamoto. Key Distribution Systems Based on Identification Information. In *Advances in Cryptology – Crypto 1987*, pages 194–202. Springer-Verlag, 1987. Vol. 293/1988 of LNCS.
25. E. Okamoto and K. Tanaka. Key Distribution System Based on Identification Information. *IEEE Journal on Selected Areas in Communications*, 7(4):481–485, May 1989.
26. T. Okamoto, R. Tso, and E. Okamoto. One-Way and Two-Party ID-based Key Agreement Protocols using Pairing. In *MDAI 2005*, Vol. 2005/2001 of LNCS, pages 122–133. Springer-Verlag, 2001.
27. E.-K. Ryu, E.-J. Yoon, and K.-Y. Yoo. An Efficient ID-Based Authenticated Key Agreement Protocol from Pairings. In *3rd International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols - NETWORKING 2004*, pages 1464–1469. Springer-Verlag, 2004. Vol. 3042/2004 of LNCS.
28. R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003. `http://eprint.iacr.org/2003/054/`.
29. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairing. In *The 2000 Sympoium on Cryptography and Information Security - SCIS 2000*, 2000.
30. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology - Crypto 1984*, pages 47–53. Springer-Verlag, 1984. Vol. 196/1985 of LNCS.
31. K. Shim. Efficient ID-based Authenticated Key Agreement Protocol based on Weil Pairing. *IEE Electronics Letters*, 39(8):653–654, 2002.
32. N. Smart. An Identity based Authenticated Key Agreement Protocol based on the Weil Pairing. *Electronics Letters*, pages 630–632, 2002.
33. H.-M. Sun and B.-T. Hsieh. Security Analysis of Shim's Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2003/113, 2003. `http://eprint.iacr.org/2003/113`.
34. Y. Wang. Efficient Identity-Based and Authenticated Key Agreement Protocol. Cryptology ePrint Archive, Report 2005/108, 2005. `http://eprint.iacr.org/2005/108/`.
35. G. Xie. An ID-Based Key Agreement Scheme from Pairing. Cryptology ePrint Archive, Report 2005/093, 2005. `http://eprint.iacr.org/2005/093/`.
36. X. Yi. An Identity-Based Signature Scheme from the Weil Pairing. *IEEE Communications Letters*, 7(2):76–78, 2003.
37. F. Zhang, R. Safavi-Naini, and W. Susilo. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In *Public Key Cryptography - PKC 2004*, pages 277–290. Springer-Verlag, 2004. Vol. 2947/2004 of LNCS.

Related-Key Differential Attacks on Cobra-S128, Cobra-F64a, and Cobra-F64b[*]

Changhoon Lee[1], Jongsung Kim[2,**], Seokhie Hong[1], Jaechul Sung[3], and Sangjin Lee[1]

[1] Center for Information Security Technologies(CIST), Korea University, Anam Dong, Sungbuk Gu, Seoul, Korea
{crypto77, hsh, sangjin}@cist.korea.ac.kr

[2] Katholieke Universiteit Leuven, ESAT/SCD-COSIC, Belgium
Kim.Jongsung@esat.kuleuven.be

[3] Department of Mathematics, University of Seoul, 90 Cheonnong Dong, Dongdaemun Gu, Seoul, Korea
jcsung@uos.ac.kr

Abstract. Data-dependent permutations (DDPs) which are very suitable for cheap hardware implementations have been introduced as a cryptographic primitive. Cobra-S128 and Cobra-F64 (which is a generic name for Cobra-F64a and Cobra-F64b) are 128-bit and 64-bit iterated block ciphers with a 128-bit key size based on such DDPs, respectively. Unlike the predecessor DDP-based ciphers [16,5], Cobra-S128 is a software-oriented cipher and Cobra-F64 is a firmware-suitable cipher. In this paper, we derive several structural properties of Cobra-S128 and Cobra-F64 and then use them to devise key recovery attacks on Cobra-S128 and Cobra-F64. These works are the first known attacks on Cobra-S128 and Cobra-F64.

Keywords: Cobra-S128, Cobra-F64, Block Cipher, Related-Key Attack, Data-Dependent Permutation.

1 Introduction

Recently, data-dependent permutations(DDPs) have been proposed as a cryptographic primitive suitable for cheap hardware implementation. For examples, CIKS-1 [16], SPECTR-H64 [5], and CIKS-128 [2] have been designed based on such DDPs. These ciphers use very simple key scheduling in order to have no time consuming key preprocessing. So, they are suitable for the applications of many network requiring high speed encryption in the case of frequent change of

[*] This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

[**] The second author was financed by Ph.D. grants of the Katholieke Universiteit Leuven and of CIST, Korea University and supported by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme under Contract IST2002507932 ECRYPT

Table 1. Summary of our related-key differential attacks

Block Cipher	Number of Rounds	Complexity Data / Time	Recovered Key Bits
Cobra-S128 (12 rounds)	12	2^{74}RK-CP / 2^{74}	6
	12	2^{74}RK-CP / 2^{122}	128(master key)
	12	2^{83}RK-CP / 2^{83}	21
	12	2^{83}RK-CP / 2^{107}	128(master key)
Cobra-F64a (16 rounds)	11	2^{59}RK-CP / 2^{107}	128(master key)
Cobra-F64b (20 rounds)	18	2^{58}RK-CP / 2^{122}	128(master key)

RK-CP: Related-Key Chosen Plaintexts, Time: Encryption units

keys. Up to now, these ciphers seem to be secure against well known attack methods such as differential cryptanalysis(DC) and linear cryptanalysis(LC) [1,15,14,11,3]. However, some researchers showed that some DDP-based ciphers with simple key schedules are vulnerable to the related-key attack [12,13].

Cobra-S128 and Cobra-F64 [4], which use a new DDP and a switchable operation, were proposed to improve the existing DDP-based ciphers. In contrast to the existing DDP-based ciphers which are based on hardware implementation, Cobra-S128 [4] is a 128-bit software-oriented cipher, and Cobra-F64 is a 64-bit firmware-suitable cipher. Note that Cobra-F64 is a generic name for Cobra-F64a and Cobra-F64b.

In this paper, we introduce structural properties for DDP-boxes used in the round function of Cobra-S128 and Cobra-F64, which allow us to make desired related-key differential characteristics. Then, we show how to exploit related-key differential characteristics to devise key recovery attacks on full-round Cobra-S128, 11-round Cobra-F64a and 18-round Cobra-F64b. See Table 1 for our results.

This paper is organized as follows; In Sect. 2, we mention some notations used in this paper and introduce several properties of DDP-boxes. Section 3 briefly describes the Cobra-S128, Cobra-F64 algorithms, and their structural properties, and Section 4 shows our related-key differential characteristics of Cobra-S128, Cobra-F64. We present key recovery attacks of Cobra-S128 and Cobra-F64 in Sect. 5. Section 6 concludes the paper.

2 Preliminaries

2.1 Notations

For convenience, we use the same notations used in [4]. Bits will be numbered from left to right, starting with bit 1. If $P = (p_1, p_2, \cdots, p_n)$ then p_1 is the most significant bit and p_n is the least significant bit.

– e_i : A binary string in which the i-th bit is one and the others are zeroes, e.g., $e_1 = (1, 0, \cdots, 0)$.

- $\oplus$: Bitwise-XOR operation
- $\boxplus(\boxminus)$ Modulo 2^{32} addition(subtraction)
- $\ggg$: Right cyclic rotation
- $Hw(A)$: Hamming weight of any binary string A

2.2 DDP-Boxes

In general, DDP-box used in DDP-based ciphers is defined as follows;

Definition 1. *Let $F(X,V)$ be the two-variable function such that $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. The function $F(X,V)$ is called a DDP-box, if for each fixed V function $F(X,V)$ is a bijective mapping defined as bit permutation.*

We denote the above DDP-box $F(X,V)$ by $P_{n/m}$ (See Fig. 1). The $P_{n/m}$-box is constructed by using elementary switching elements $P_{2/1}$ as elementary building blocks performing controlled transposition of two input bits x_1 and x_2. Here, $P_{2/1}$-box is controlled with one bit v and outputs two bits y_1 and y_2, where $y_1 = x_{1+v}$ and $y_2 = x_{2-v}$, i.e., if $v = 1$, it swaps two input bits otherwise (if $v = 0$), does not.

In other words, $P_{n/m}$-box can be represented as a superposition of the operations performed on bit sets :

$$P_{n/m} = L^{V_1} \circ \pi_1 \circ L^{V_2} \circ \pi_2 \circ \cdots \circ \pi_{s-1} \circ L^{V_s}$$

where L is an active layer composed of $n/2$ $P_{2/1}$ parallel elementary boxes, $V_1, V_2, \cdots V_s$ are control vectors of the active layers from 1 to $s = 2m/n$, and $\pi_1, \pi_2, \cdots, \pi_{s-1}$ are fixed permutations (See Fig. 1). Fig. 2 shows structure of the $P_{32/96}$ and $P^{-1}_{32/96}$ used in Cobra-S128 and Cobra-F64. Due to the symmetric structure, the mutual inverses, $P_{32/96}$ and $P^{-1}_{32/96}$, differ only with the distribution of controlling bits over the boxes $P_{2/1}$, i.e., $P^{V}_{32/96}$ and $P^{V'}_{32/96}$ are mutually inverse when $V = (V_1, V_2, \cdots, V_6)$ and $V' = (V_6, V_5, \cdots, V_1)$.

Now, we introduce some properties of DDP-boxes. We let x_1x_2 be a two-bit input string of $P_{2/1}$ and v be an one-bit control vector.

Property 1. [12,13] $P_{2/1(v=0)}(x_1x_2) = P_{2/1(v=1)}(x_1x_2)$ with probability 2^{-1}.

The equation in the above property holds only when $x_1 = x_2$.

Property 2. [12,13] Let an input and control vector differences of $P_{2/1}$-box be $\Delta X = X \oplus X'$ and $\Delta V = V \oplus V'$ respectively, where X and X' are two-bit input vectors, and V and V' are one-bit control vectors. Then we have the following equations.

a) If $\Delta X = 10$ or 01, and $\Delta V = 0$ then the corresponding output difference of $P_{2/1}$-box, ΔY, is 10 with probability 2^{-1} or 01 with probability 2^{-1}.
b) If $\Delta X = 10$ or 01 and $\Delta V = 1$ then the corresponding output difference of $P_{2/1}$-box, ΔY, is 10 with probability 2^{-1} or 01 with probability 2^{-1}.

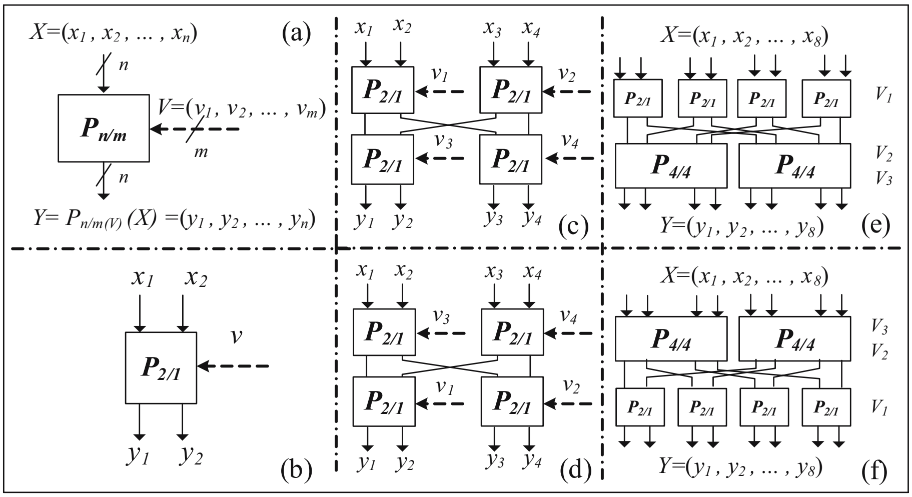


Fig. 1. CP-boxes : (a) $P_{n/m}$, (b) $P_{2/1}$, (c) $P_{4/4}$, (d) $P^{-1}_{4/4}$, (e) $P_{8/12}$, (f) $P^{-1}_{8/12}$

c) If $\Delta X = 00$ and $\Delta V = 1$ such that $X = X' = 10$ or 01 then the corresponding output difference of $P_{2/1}$-box is $\Delta Y = 11$. Thus, if $\Delta X = 00$ and $\Delta V = 1$ then the corresponding output difference of $P_{2/1}$-box is $\Delta Y = 11$ with probability 2^{-1}.

The above properties are also expanded into the following properties.

Property 3. [12,13] Let V and V' be m-bit control vectors for $P_{n/m}$-box such that $V \oplus V' = e_i$ $(1 \le i \le m)$. Then $P_{n/m(V)}(X) = P_{n/m(V')}(X)$ with a probability of 2^{-1} where $X \in \{0,1\}^n$. It also holds in $P^{-1}_{n/m}$-box.

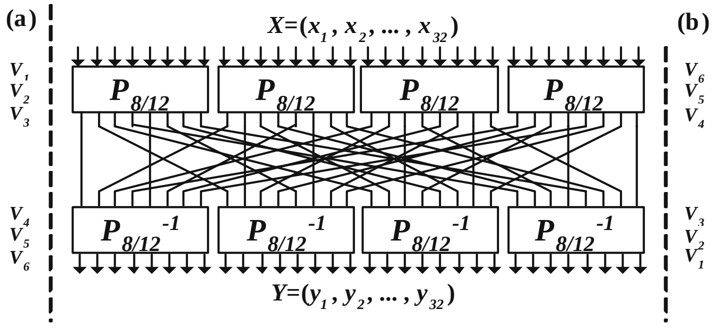


Fig. 2. CP-boxes : (a) $P_{32/96}$, (b) $P^{-1}_{32/96}$

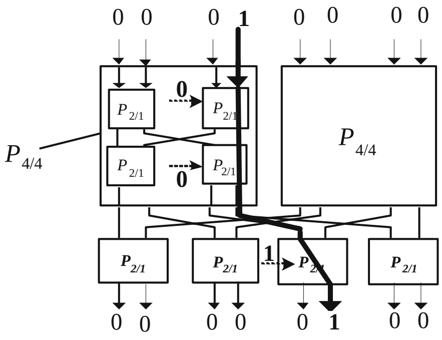


Fig. 3. An example of the difference route when the input and output differences of $P^{-1}_{8/12}$-box are fixed as e_4 and e_6, respectively

Property 4. [12,13] If $X \oplus X' = e_i$ $(1 \le i \le m)$ then $P_{8/12(V)}(X) \oplus P_{8/12(V)}(X')$ $= e_j$ for some j $(1 \le j \le m)$. In addition, if i and j are fixed then the exact difference route from i to j via three $P_{2/1}$-boxes is also fixed. It also holds in $P^{-1}_{8/12}$-box.

For example, consider $i = 4$ and $j = 6$ in the *Property* 4. Then, we can exactly know the 3 bits of control vectors (0,0,1) corresponding to three elements $P_{2/1}$-boxes of $P^{-1}_{8/12}$-box with probability 1. See Fig. 3. In Fig. 3, the bold line denotes the difference route when the input and output differences of $P^{-1}_{8/12}$-box are fixed as e_4 and e_6, respectively.

Property 5. [12,13] Let $Y = P_{n/m(V)}(X)$ and $Y' = P_{n/m(V)}(X')$. Then $Hw(X \oplus X') = Hw(Y \oplus Y')$. It also holds in $P^{-1}_{n/m}$-box.

3 Cobra-S128 and Cobra-F64

In this section, we briefly describe the block cipher Cobra-S128, Cobra-F64a, and Cobra-F64b [4] and derive their several properties used in our attacks.

3.1 Description of Cobra-S128 Cipher

Cobra-S128 is a 128-bit iterated block cipher with a 128-bit key size and 12 rounds. This cipher is composed of the initial transformation, e-dependent round function $Crypt^{(e)}$, and the final transformation where $e = 0 (e = 1)$ denotes encryption(decryption). The data encryption procedure is performed as follows. See Fig. 6 in Appendix A.

1. An 128-bit input data block is divided into four 32-bit subblocks A, B, C, D.
2. Perform initial transformation :
   $(A, B, C, D) := (A \oplus Q_1^{(1,e)}, B \oplus Q_2^{(1,e)}, C \oplus Q_3^{(1,e)}, D \oplus Q_4^{(1,e)})$

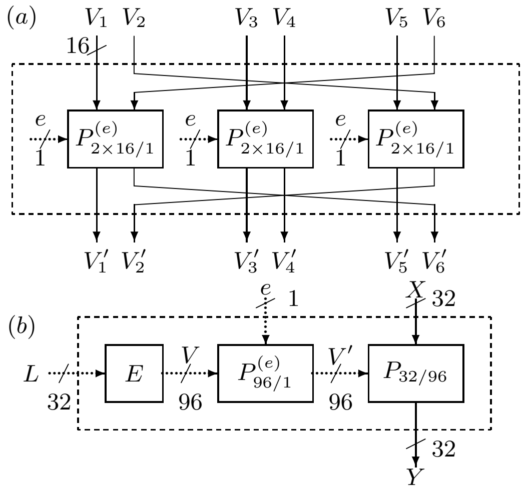


Fig. 4. (a) $P^{(e)}_{96/1}$, (b) $P^{(e)}_{32/32}$

3. For $j = 1$ to 11 do :
   $(A, B, C, D) := Crypt^{(e)}(A, B, C, D, Q^{(1,e)}_j, Q^{(2,e)}_j)$; $(A, B, C, D) := (B, A, D, C)$
4. $j = 12$ do :
   $(A, B, C, D) := Crypt^{(e)}(A, B, C, D, Q^{(1,e)}_{12}, Q^{(2,e)}_{12})$;
5. Perform final transformation :
   $(A, B, C, D) := (A \oplus Q^{(2,e)}_{12}, B \oplus Q^{(2,e)}_{11}, C \oplus Q^{(2,e)}_{10}, D \oplus Q^{(2,e)}_{9})$
6. Output (A, B, C, D)

The $Crypt^{(e)}$ function is composed of the basic arithmetical operations $(\oplus, \boxplus, \boxminus)$ and DDP-box $P_{32/32}$ which is made up of a extension box E, a simple transposition box $P^{(e)}_{96/1}$, and $P_{32/96}$ (See Fig. 4). The extension box E provides the following relation between its input $L = (l_1, \cdots, l_{32})$ and output $V = (V_1, \cdots, V_6)$:

$$V_1 = L_l,\ V_2 = L_l^{\ggg 6},\ V_3 = L_l^{\ggg 12},\ V_4 = L_r,\ V_5 = L_r^{\ggg 6},\ V_6 = L_r^{\ggg 12}$$

where $L_l = (l_1, \cdots, l_{16})$, $L_r = (l_{17}, \cdots, l_{32})$, $|l_i| = 1$ $(1 \leq i \leq 32)$ and $|V_i| = 16$ $(1 \leq i \leq 6)$. The transposition box $P^{(e)}_{96/1}$ is implemented as some single layer controlled permutation box consisting of three parallel single layer boxes $P^{(e)}_{2\times16/1}$ (See Fig. 4). An input of each $P^{(e)}_{2\times16/1}$-box is divided into 16-bit left and 16-bit right inputs, and contains 16 parallel $P^{(e)}_{2/1}$-boxes controlled with the same bit e. So, if the input vector of the box $P^{(e)}_{96/1}$ is $V = (V_1, \cdots, V_6)$ then the corresponding output vector is $V' = (V_1, \cdots, V_6)$ when $e = 0$ or $V' = (V_6, \cdots, V_1)$ when $e = 1$.

Table 2. Key schedule of Cobra-S128, Cobra-F64a, and Cobra-F64b

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	.
$Q_j^{(1,0)}$	K_1	K_2	K_3	K_4	K_2	K_1	K_4	K_3	K_1	K_2	K_4	K_3	K_1	K_4	K_2	K_3	K_2	K_4	K_3	K_1	K_2	.
$Q_j^{(2,0)}$	K_4	K_3	K_1	K_2	K_3	K_2	K_1	K_4	K_2	K_3	K_1	K_2	K_3	K_1	K_3	K_4	K_3	K_1	K_4	K_2	K_3	.

The key schedule of Cobra-S128 is very simple. An 128-bit master key K is split into four 32-bit blocks ,i.e., $K = (K_1, K_2, K_3, K_4)$. Then, in order to generate the subkey sequences $(Q_j^{(1,0)}, Q_j^{(2,0)})$, K_1,K_2,K_3 and K_4 are rearranged as specified in Table 2 in which $(Q_j^{(1,0)}, Q_j^{(2,0)})$ denotes the j-th round key sequence $(1 \le j \le 12)$, and $Q_j^{(1,0)}, Q_j^{(2,0)} \in \{0,1\}^{32}$.

3.2 Description of Cobra-F64 Ciphers

Cobra-F64a and Cobra-F64b, which are suitable for firmware implementation, are 64-bit iterated block ciphers with a 128-bit key size and 16 and 20 rounds, respectively. These ciphers perform data encryption procedure as follows;

1. 64-bit input data block is divided into two 32-bit subblocks A, B.
2. For $j = 1$ to $R - 1$ do :
   $(A, B) := Crypt^{(e)}(A, B, Q_j^{(1,e)}, Q_j^{(2,e)})$; $(A, B) := (B, A)$
3. $j = R$ do :
   $(A, B) := Crypt^{(e)}(A, B, Q_j^{(1,e)}, Q_j^{(2,e)})$;
4. Perform final transformation :
   $(A, B) := (A \oplus Q_{R+1}^{(1,e)}, B \oplus Q_{R+1}^{(2,e)})$ for Cobra-F64b and $(A, B) := (A \boxminus Q_{R+1}^{(1,e)}, B \boxplus Q_{R+1}^{(2,e)})$ for Cobra-F64a
5. Output $(A \parallel B)$

The detailed description for $Crypt^{(e)}$ is presented in Fig. 7 in Appendix A. Cobra-F64a and Cobra-F64b also use Table. 2 as key schedule.

3.3 Properties of Cobra-S128 and Cobra-F64

In this subsection, we derive some properties of operations used in round function Cobra-S128 and Cobra-F64 which are useful to construct related-key differential characteristics.

Property 6. Let ΔX and ΔV be differences of input and control vector of $P_{32/96}$, respectively. Then we can get the following properties of $P_{32/96}$ from the definition of $P_{32/96}$ and the previous properties (*Property* 1,2,3,5).

a) $\Delta P_{32/96(\Delta V=0)}(\Delta X=0)=0$ with a probability of 1.
b) $\Delta P_{32/96(\Delta V=e_1)}(\Delta X=0)=0$ with a probability of 2^{-1}.
c) $\Delta P_{32/96(\Delta V=0)}(\Delta X=e_1)=e_1$ with a probability of 2^{-6} because $P_{32/96}$ consists of 6 active layers.
d) $\Delta P_{32/96(\Delta V=e_1)}(\Delta X=e_1)=e_1$ with a probability of 2^{-6} because $P_{32/96}$ consists of 6 active layers.

Similarly, we can also derive the difference property for $P_{32/32}$ as follows.

Property 7. Let ΔX and ΔV be differences of input and control vector of $P_{32/32}$, respectively. Then the following equations are obtained from the definition of extension box E used in $P_{32/32}$ and the above *Property* 6.

a) $\Delta P_{32/32(\Delta L=0)}(\Delta X=0)=0$ with probability 1.
b) $\Delta P_{32/32(\Delta L=e_1)}(\Delta X=0)=0$ with probability 2^{-3}.
c) $\Delta P_{32/32(\Delta L=0)}(\Delta X=e_1)=e_1$ with probability 2^{-6}.
d) $\Delta P_{32/32(\Delta L=e_1)}(\Delta X=e_1)=e_1$ with probability 2^{-8}.
e) $\Delta P_{32/32(\Delta L=e_9)}(\Delta X=e_1)=e_1$ with probability 2^{-9}.
f) $\Delta P_{32/32(\Delta L=e_{1,9})}(\Delta X=e_1)=e_1$ with probability 2^{-11}.
g) $HW(\Delta P_{32/32(\Delta L=e_1)}(\Delta X=0))=0,2,4,6$ by *Property* 2, 5.

4 Related-Key Differential Characteristics of Cobra-S128 and Cobra-F64

In this section, we construct related-key differential characteristics for Cobra-S128 and Cobra-F64 using the properties mentioned in the previous subsection.

As stated earlier, the key schedules of the Cobra-S128 and Cobra-F64 are very simple, i.e., the round keys are only 32-bit parts of the 128-bit master key, and there are many properties due to the structural feature of $P_{32/32}$-box. They allow us to construct good related-key differential characteristics even though it uses an $P_{32/32}$-box.

In order to find good related-key differential characteristics, we performed a series of simulations (in which we used a number of plaintext and key differences whose hamming weights are one in each 32-bit word). As our simulation results, we obtained a full-round (12 rounds) related-key differential characteristic $(0,0,e_1,e_1)\rightarrow(0,0,0,t)$ of Cobra-S128 with probability 2^{-72}, a 12-round related-key differential characteristic $(e_1,e_1)\rightarrow(e_1,t)$ of Cobra-F64a with probability 2^{-62}, and a full-round (20 rounds) related-key differential characteristic $(0,e_1)\rightarrow(e_1\oplus t^{\ggg 8},t)$ of Cobra-F64b with probability 2^{-62}, where t represents any 32-bit word of which the first byte have hamming weight 1 and the second byte has also hamming weight 1 and the other two bytes has hamming weight 0. Note that these related-key differential characteristics of Cobra-S128, Cobra-F64a, and Cobra-F64b include their final transformations(FT) and use key differences $(0,0,e_1,0)$, $(0,0,0,e_1)$, and (e_1,e_1,e_1,e_1), respectively. Subsection 4.1 describes our full-round related-key differential characteristic of Cobra-S128 in more detail. See appendix B for the complete forms of characteristics of

Cobra-F64a and Cobra-F64b (which can be constructed by the same arguments as in the below subsection). But, in our attacks, we use 11-round related-key differential characteristic of Cobra-F64a and 18-round related-key differential characteristic of Cobra-F64b because the attacks over 11-round Cobra-F64a and 18-round Cobra-F64b lead more complexities than exhaustive search.

4.1 How to Construct the Full-Round Related-Key Differential Characteristic of Cobra-S128

In Table 3, ΔRI^{IT} and ΔRK^{IT} denote the plaintext and initial key differences, respectively. ΔRI^{i} and ΔRK^{i} are the input and key differences of *ith* round, respectively, and ($P1$,$P2$,$P3$,$P4$,$P5$,$P6$) are the respective probabilities that for given input and control vector differences of ($P^{B,e}_{32/32}$, $P^{C,1}_{32/32}$, $P^{C,e}_{32/32}$, $P^{C',e}_{32/32}$, $P^{B,0}_{32/32}$, $P^{B',e}_{32/32}$) boxes used in Cobra-S128, their corresponding output differences satisfy a specific output difference, e_1 or 0. These probabilities are obtained from *Property* 6, 7.

Specifically, since the plaintext and initial key differences of Cobra-S128 are $(0, 0, e_1, e_1)$ and $(0, 0, e_1, 0)$ respectively, the input difference of the first round is to be $(0, 0, 0, e_1)$. Thus the output differences of $P^{B,e}_{32}$ and $P^{C,1}_{32}$ are 0 because the input and control vector differences of $P^{B,e}_{32/32}$ and $P^{C,1}_{32/32}$ are 0. Also, since the input and control vector differences of $P^{C,e}_{32/32}$ are e_1 and 0, respectively, the corresponding output difference of $P^{C,e}_{32/32}$ is to be e_1 with probability 2^{-6} by *Property* 6, 7. Similarly, the output differences of $P^{B,0}_{32}$ and $P^{C',e}_{32}$ are 0 with the probability of 1 because the input and control vector differences of $P^{B,0}_{32/32}$ and $P^{C',e}_{32/32}$ are 0. Furthermore, since the input and control vector differences of $P^{B',e}_{32/32}$ are e_1 and 0 in the first round, respectively, the corresponding output difference of $P^{B',e}_{32/32}$ is to be e_1 with a probability of 2^{-6} by *Property* 6, 7. So, the output difference of the first round is $(0, 0, 0, e_1)$ with a probability of 2^{-12}, i.e., the input difference of the second round is $(0, 0, e_1, 0)$ as mentioned in Table 3. Repeating this manner for the rest of the rounds, we can obtain an output difference $(0, 0, 0, 0)$ after 11 rounds with probability 2^{-66} as presented in Table 3.

Proceeding to the last round, since the input and key difference are $(0, 0, 0, 0)$ and $(e_1, 0)$, respectively, the input and control vector differences of $P^{(B,0)}_{32/32}$ are 0 and e_1, respectively, and thus the corresponding output difference of $P^{(B,0)}_{32/32}$ is to be 0 with probability 2^{-3}. So, the input and control vector differences of $P^{(B',e)}_{32/32}$ are 0 and e_1, respectively. Here, we consider the hamming weight of the output difference of $P^{(B',e)}_{32/32}$ in the last round in order to construct a related-key differential characteristic suitable for our attack scenario. The hamming weight of the output difference of $P^{(B',e)}_{32/32}$ depends on an input form of $P_{32/96}$ in $P^{B',e}_{32/32}$. Let the control vector of $P_{32/96}$ in $P^{B',e}_{32/32}$ be $V' = (V'_1, V'_2, V'_3, V'_4, V'_5, V'_6)$. Since

Table 3. Related-Key Differential Characteristic of Cobra-S128

Round (i)	ΔRI^i	ΔRK^i	$P1/P2/P3/P4/P5/P6$	Prob.
IT	$(0,0,e_1,e_1)$	$(0,0,e_1,0)$	·	1
1	$(0,0,0,e_1)$	$(0,0)$	$1/1/2^{-6}/1/1/2^{-6}$	2^{-12}
2	$(0,0,e_1,0)$	$(0,e_1)$	$1/2^{-3}/2^{-3}/1/1/1$	2^{-6}
3	$(0,0,0,0)$	$(e_1,0)$	$1/1/1/1/2^{-3}/2^{-3}$	2^{-6}
4	$(0,0,0,e_1)$	$(0,0)$	$1/1/2^{-6}/1/1/2^{-6}$	2^{-12}
5	$(0,0,e_1,0)$	$(0,e_1)$	$1/2^{-3}/2^{-3}/1/1/1$	2^{-6}
6	$(0,0,0,0)$	$(0,0)$	$1/1/1/1/1/1$	1
7	$(0,0,0,0)$	$(0,0)$	$1/1/1/1/1/1$	1
8	$(0,0,0,0)$	$(e_1,0)$	$1/1/1/2^{-3}/2^{-3}$	2^{-6}
9	$(0,0,0,e_1)$	$(0,0)$	$1/1/2^{-6}/1/1/2^{-6}$	2^{-12}
10	$(0,0,e_1,0)$	$(0,e_1)$	$1/2^{-3}/2^{-3}/1/1/1$	2^{-6}
11	$(0,0,0,0)$	$(0,0)$	$1/1/1/1/1/1$	1
12	$(0,0,0,0)$	$(e_1,0)$	$1/1/1/1/2^{-3}/2^{-3}$	2^{-6}
FT	$(0,0,e_1,t)$	$(0,0,e_1,0)$	·	1
Output	$(0,0,0,t)$	·	·	·
Total	·	·	·	2^{-72}

the control vector difference of $P^{B',e}_{32/32}$ is e_1, it is propagated via E into the first bit of V'_1, the seventh bit of V'_2, and 13th bit of V'_3 of $P_{32/96}$. For convenience, we denote three $P_{2/1}$-boxes corresponding to the first bit of V'_1, the seventh bit of V'_2, and 13th bit of V'_3 $P^{V'_{1_1}}_{2/1}$, $P^{V'_{2_7}}_{2/1}$, and $P^{V'_{3_{13}}}_{2/1}$, respectively (See Fig. 5). Note that an input difference of $P_{32/96}$ in $P^{B',e}_{32/32}$ is 0. Then we can classify the input forms of these $P_{2/1}$-boxes corresponding to the above 3 controlled bits into 8 cases. However, in our attack, we consider the hamming weight of the output difference of $P^{(B',e)}_{32/32}$ in the last round is to be 2 under the condition that the form of input pair $(x_1x_2, x'_1x'_2)$ of $P^{V'_{3_{13}}}_{2/1}$ has $(10,10)$ or $(01,01)$, and the input pair of $P^{V'_{1_1}}_{2/1}$ and $P^{V'_{2_7}}_{2/1}$ has any value $(x_1x_2, x'_1x'_2)$ whose difference is zero. Then the output difference of $P^{V'_{3_{13}}}_{2/1}$ has 11 with probability 2^{-1} by *Property* 2-c), and the output differences of $P^{V'_{1_1}}_{2/1}$ and $P^{V'_{2_7}}_{2/1}$ have 00 with a probability of 2^{-1} by *Property* 1, respectively. In more detail, one-bit of two-bit active output differences of $P^{V'_{3_{13}}}_{2/1}$ is propagated into the fourth-bit of the first $P^{-1}_{8/12}$ in $P_{32/96}$ and the other one-bit is propagated into the fourth-bit of the second $P^{-1}_{8/12}$ in $P_{32/96}$ (Refer to Fig. 5). So the resultant probability is $2^{-3}(= P_6)$ that satisfies the first and second output bytes of $P_{32/32}$ having hamming weight 1 and the other bytes having hamming weight 0 by *Property* 5, 7. Hence a full-round related-key differential characteristic $(0,0,e_1,e_1) \rightarrow (0,0,0,t)$ holds with probability 2^{-72} when $K \oplus K' = (0,0,e_1,0)$.

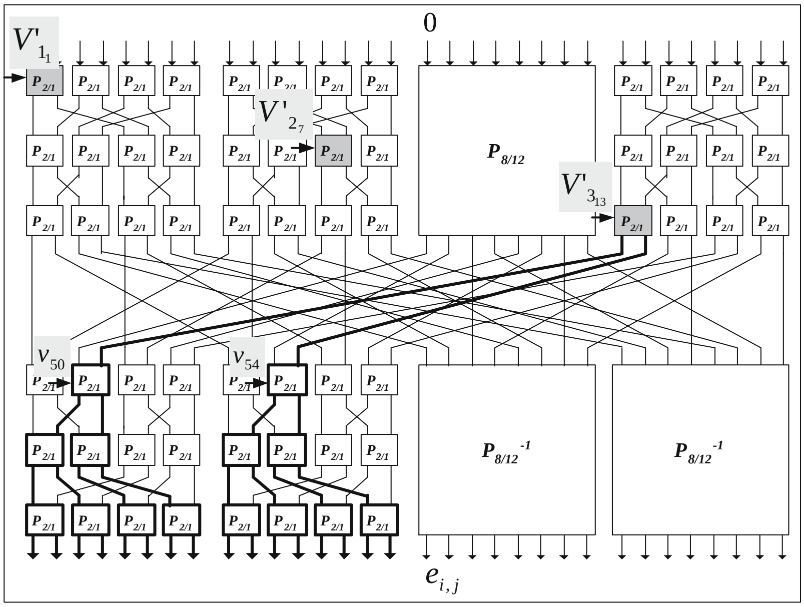


Fig. 5. The possible routes of the output difference of $P_{2/1}^{V_3'13}$-box in $P_{32/96}$

5 Related-Key Differential Attacks on Cobra-S128 and Cobra-F64

We now present key recovery attacks on Cobra-S128 and Cobra-F64 using our related-key differential characteristics.

5.1 Attack Procedure on Cobra-S128

We first show how to search for a master key pair of Cobra-S128 by using the full-round related-key differential characteristic presented in Section 4. Note that, from the previous full-round related-key differential characteristic, we know that the hamming weight of output difference of $P_{32/32}^{B',e}$ in the last round is 2 (one is in the first $P_{8/12}^{-1}$ and the other is in the second $P_{8/12}^{-1}$) with probability 2^{-72}.

To begin with, we encrypt 2^{73} plaintext pairs $P = (P_{LL}, P_{LR}, P_{RL}, P_{RR})$ and $P' = (P_{LL}, P_{LR}, P_{RL} \oplus e_1, P_{RR} \oplus e_1)$ under an unknown key $K = (K_1, K_2, K_3, K_4)$ and an unknown related-key $K' = (K_1, K_2, K_3 \oplus e_1, K_4)$, respectively, and then get the 2^{73} corresponding ciphertext pairs $C = (C_{LL}, C_{LR}, C_{RL}, C_{RR})$ and $C' = (C'_{LL}, C'_{LR}, C'_{RL}, C'_{RR})$, i.e., $E_K(P) = C$ and $E_{K'}(P) = C'$, where E is the block cipher Cobra-S128. Since our full-round related-key differential characteristic of Cobra-S128 has a probability of 2^{-72}, we expect at least one ciphertext pair (C, C') such that $C \oplus C' = (0, 0, 0, t)$ with a probability of $1-(1-2^{-72})^{2^{73}} \approx 0.87$.

According to our differential trail described in Table 3, we can deduce that the two difference bits in such (C, C') are derived from $V'_{3_{13}}$ of the last $P_{32/96}$ (Refer to Fig 5). That is, we can expect that there are two exact routes: One is from the $4th$ bit in input difference of the first $P^{-1}_{8/12}$ to the ith bit of the output difference of the first $P^{-1}_{8/12}$ $(1 \le i \le 8)$ and the other is from the $12th$ bit in input difference of the second $P^{-1}_{8/12}$ to the ith bit of the output difference of the second $P^{-1}_{8/12}$ $(9 \le i \le 16)$. Note that the control vectors and key bits corresponding to the above routes are uniquely determined by Property 4. See Figs. 3 and 5.

Table 4. Classes of the control vectors and key bits corresponding to the possible routes when the fourth bit of the first $P^{-1}_{8/12}$ and output difference e_i in $P_{32/96}$-box are fixed

Class	e_i	Control vectors	Key bits
$\mathcal{CL}_1$	e_1	$v_{50} = C^{19}_{RL} \oplus K^{19}_3 = 1,\ v_{65} = C^{27}_{RL} \oplus K^{27}_3 = 1,\ v_{81} = C^{21}_{RL} \oplus K^{21}_3 = 0$	$K^{19}_3, K^{27}_3, K^{21}_3$
	e_2	$v_{50} = C^{19}_{RL} \oplus K^{19}_3 = 1,\ v_{65} = C^{27}_{RL} \oplus K^{27}_3 = 1,\ v_{81} = C^{21}_{RL} \oplus K^{21}_3 = 1$	
$\mathcal{CL}_2$	e_3	$v_{50} = C^{19}_{RL} \oplus K^{19}_3 = 1,\ v_{65} = C^{27}_{RL} \oplus K^{27}_3 = 0,\ v_{82} = C^{22}_{RL} \oplus K^{22}_3 = 0$	$K^{19}_3, K^{27}_3, K^{22}_3$
	e_4	$v_{50} = C^{19}_{RL} \oplus K^{19}_3 = 1,\ v_{65} = C^{27}_{RL} \oplus K^{27}_3 = 0,\ v_{82} = C^{22}_{RL} \oplus K^{22}_3 = 1$	
$\mathcal{CL}_3$	e_5	$v_{50} = C^{19}_{RL} \oplus K^{19}_3 = 0,\ v_{66} = C^{28}_{RL} \oplus K^{28}_3 = 1,\ v_{83} = C^{23}_{RL} \oplus K^{23}_3 = 0$	$K^{19}_3, K^{28}_3, K^{23}_3$
	e_6	$v_{50} = C^{19}_{RL} \oplus K^{19}_3 = 0,\ v_{66} = C^{28}_{RL} \oplus K^{28}_3 = 1,\ v_{83} = C^{23}_{RL} \oplus K^{23}_3 = 1$	
$\mathcal{CL}_4$	e_7	$v_{50} = C^{19}_{RL} \oplus K^{19}_3 = 0,\ v_{66} = C^{28}_{RL} \oplus K^{28}_3 = 0,\ v_{84} = C^{24}_{RL} \oplus K^{24}_3 = 0$	$K^{19}_3, K^{28}_3, K^{24}_3$
	e_8	$v_{50} = C^{19}_{RL} \oplus K^{19}_3 = 0,\ v_{66} = C^{28}_{RL} \oplus K^{28}_3 = 0,\ v_{84} = C^{24}_{RL} \oplus K^{24}_3 = 1$	

Table 5. Classes of the control vectors and key bits corresponding to the possible routes when the fourth bit of the second $P^{-1}_{8/12}$ and output difference e_i in $P_{32/96}$-box are fixed

Class	e_i	Control vectors	Key bits
$\mathcal{CL}_5$	e_9	$v_{54} = C^{23}_{RL} \oplus K^{23}_3 = 1,\ v_{69} = C^{31}_{RL} \oplus K^{31}_3 = 1,\ v_{85} = C^{25}_{RL} \oplus K^{25}_3 = 0$	$K^{23}_3, K^{31}_3, K^{25}_3$
	e_{10}	$v_{54} = C^{23}_{RL} \oplus K^{23}_3 = 1,\ v_{69} = C^{31}_{RL} \oplus K^{31}_3 = 1,\ v_{85} = C^{25}_{RL} \oplus K^{25}_3 = 1$	
$\mathcal{CL}_6$	e_{11}	$v_{54} = C^{23}_{RL} \oplus K^{23}_3 = 1,\ v_{69} = C^{31}_{RL} \oplus K^{31}_3 = 0,\ v_{86} = C^{26}_{RL} \oplus K^{26}_3 = 0$	$K^{23}_3, K^{31}_3, K^{26}_3$
	e_{12}	$v_{54} = C^{23}_{RL} \oplus K^{23}_3 = 1,\ v_{69} = C^{31}_{RL} \oplus K^{31}_3 = 0,\ v_{86} = C^{26}_{RL} \oplus K^{26}_3 = 1$	
$\mathcal{CL}_7$	e_{13}	$v_{54} = C^{23}_{RL} \oplus K^{23}_3 = 0,\ v_{70} = C^{32}_{RL} \oplus K^{32}_3 = 1,\ v_{87} = C^{27}_{RL} \oplus K^{27}_3 = 0$	$K^{23}_3, K^{32}_3, K^{27}_3$
	e_{14}	$v_{54} = C^{23}_{RL} \oplus K^{23}_3 = 0,\ v_{70} = C^{32}_{RL} \oplus K^{32}_3 = 1,\ v_{87} = C^{27}_{RL} \oplus K^{27}_3 = 1$	
$\mathcal{CL}_8$	e_{15}	$v_{54} = C^{23}_{RL} \oplus K^{23}_3 = 0,\ v_{70} = C^{32}_{RL} \oplus K^{32}_3 = 0,\ v_{88} = C^{28}_{RL} \oplus K^{28}_3 = 0$	$K^{23}_3, K^{32}_3, K^{28}_3$
	e_{16}	$v_{54} = C^{23}_{RL} \oplus K^{23}_3 = 0,\ v_{70} = C^{32}_{RL} \oplus K^{32}_3 = 0,\ v_{88} = C^{28}_{RL} \oplus K^{28}_3 = 1$	

Fig. 5 represents the possible routes of the output difference of $P^{V'_{3_{13}}}_{2/1}$-box in $P_{32/96}$ and the bold line denotes a trace of non-zero difference. Tables 4 and 5 represent classes of the control vectors and key bits corresponding to the possible routes when i is fixed. For example, assume that output difference is $e_{1,9}$. Then since the control vectors of three $P_{2/1}$ corresponding to route from the fourth bit of input difference of the first $P^{-1}_{8/12}$ to the first bit of output difference of the first

$P_{8/12}^{-1}$ are $v_{50} = C_{RL}^{19} \oplus K_3^{19} = 1$, $v_{65} = C_{RL}^{27} \oplus K_3^{27} = 1$, and $v_{81} = C_{RL}^{21} \oplus K_3^{21} = 0$ where C^j and K^j mean the $j-th$ bits of C and K, respectively, we can know three key bits K_3^{19}, K_3^{27}, and K_3^{21}. Similarly, since the control vectors of three $P_{2/1}$ corresponding to route from the fourth bit of input difference of the second $P_{8/12}^{-1}$ to the first bit of output difference of the second $P_{8/12}^{-1}$ are $v_{54} = C_{RL}^{23} \oplus K_3^{23} = 1$, $v_{69} = C_{RL}^{31} \oplus K_3^{31} = 1$, and $v_{85} = C_{RL}^{25} \oplus K_3^{25} = 0$, we can know three key bits K_3^{23}, K_3^{31}, and K_3^{25}. Based on this idea we can devise a related-key differential attack on full-round Cobra-S128 as follows.

1. Prepare 2^{73} plaintext pairs (P_i, P_i'), $i = 1, \cdots, 2^{73}$, which have the $(0, 0, e_1, e_1)$ difference. All P_i are encrypted using a master key K and all P_i' are encrypted using a master key K' where K and K' have the $(0, 0, e_1, 0)$ difference. Encrypt each plaintext pair (P_i, P_i') to get the corresponding ciphertext pair (C_i, C_i').
2. Check that $C_i \oplus C_i' = (0, 0, 0, t)$ for each i. We call the bit positions of t whose values are 1 BOP(Bit One Position). Note that there are two BOPs.
3. For each ciphertext pair (C_i, C_i') passing Step 2, extract some bits of control vector by chasing a difference route between the first BOP and the position of the fourth input bit in the first $P_{8/12}^{-1}$ and by chasing a difference route between the second BOP and the position of the fourth input bit of the second $P_{8/12}^{-1}$. Find the corresponding bits of K_3 and K_3' by using Tables 4 and 5.

The data complexity of this attack is 2^{74} related-key chosen plaintexts. Step 1 is the data collection step and thus this step requires a time complexity of 2^{74} encryptions. By our related-key differential characteristic each ciphertext pair can pass Step 2 with probability at least 2^{-72} and thus the expectation of ciphertext pairs that pass this test is at least 2. Step 2 can be done efficiently by checking ciphertext differences in byte unit and Step 3 also requires a small amount of time complexity. Furthermore, for each ciphertext pair that passes this test the probability that its difference is derived from the $P_{2/1}^{V_{1_1}'}$ or $P_{2/1}^{V_{2_7}'}$ not $P_{2/1}^{V_{3_{13}}'}$ is less than 2^{-74}. Hence we can retrieve some portion (at least 6 bits) of subkey matrials by performing Step 3 with high probability. Moreover, if we perform an exhaustive search of the remaining key bits we can find the whole of master key pair (K, K') with a data complexity of 2^{74} related-key chosen plaintexts and a time complexity of at most 2^{122} encryptions.

From now on, we introduce improved procedure to search for a master key pair, which has a trade-off in data and time complexities. Unlike the above attack, this attack simultaneously consider three types of ciphertext pairs whose differences have hamming weight 2: the first type is associated with $V_{3_{13}}'$ as like the above attack, the second type is associated with V_{2_7}' and the third type is associated with V_{1_1}'. In this attack, we classify these three types of ciphertext pairs into Cases 1, 2, and 3, respectively.

1. Prepare 2^{82} plaintext pairs (P_i, P'_i), $i = 1, \cdots, 2^{82}$, which have the same conditions as the above Step 1. Encrypt each plaintext pair (P_i, P'_i) to get the corresponding ciphertext pair (C_i, C'_i).
2. Check that $C_i \oplus C'_i = (0, 0, 0, t')$ for each i, where t' is any 32-bit word which has hamming weight 2 such that one is in the first byte and the other is in the second byte (Case 1), or one is in the first byte and the other is in the third byte (Case 2), or one is in the third byte and the other is in the fourth byte (Case 3). We call the bit position which has 1 in t' BOP$'$.
3. For each ciphertext pair (C_i, C'_i) in Case 1, extract the corresponding 3 bits of control vector by chasing a difference route between the first BOP$'$ and the position of the fourth input bit in the first $P^{-1}_{8/12}$ and also extract the corresponding 3 bits of control vector by making a difference route between the second BOP$'$ and the position of the fourth input bit of the second $P^{-1}_{8/12}$. Compute candidates of the corresponding bits of K_3 and K'_3 by using Tables 4 and 5. Output each 3-bit subkey pair with maximal number of hits (each 3-bit subkey pair corresponds to each difference route).
4. The same arguments can be applied to the Cases 2 and 3. For each ciphertext pair (C_i, C'_i) in Case 2 (resp., Case 3), extract the corresponding 4 (resp., 5) bits of control vector by making a difference route between the first BOP$'$ and the position of the sixth input bit in the first $P^{-1}_{8/12}$ (resp., the position of the first input bit in the third $P^{-1}_{8/12}$) and also extract the corresponding 4 (resp., 5) bits of control vector by making a difference route between the second BOP$'$ and the position of the sixth input bit of the third $P^{-1}_{8/12}$ (resp., the position of the fifth input bit in the fourth $P^{-1}_{8/12}$). Compute candidates of the corresponding bits of K_3 and K'_3 by using C_i, C'_i, and extracted control vectors (in Case 2 (resp., Case 3), we can extract some bits of control vectors by making difference routes from the $P_{2/1}$ of V'_{2_7} (resp., V'_{1_1}) to BOP$'$s). Output each 4-bit (resp., 5-bit) subkey pair with maximal number of hits in Case 2 (resp., in Case 3).

The data complexity of this attack is 2^{83} related-key chosen plaintexts. The probability that a fixed two-bit difference in Case 1 is connected with the $P_{2/1}$ of $V'_{3_{13}}$ is 2^{-6} (this probability is derived from *Property* 2-*a*) and *Property* 4) and thus the total probability is $2^{-72-6} = 2^{-78}$. It follows that the expected number of hits for each right 3-bit subkey is about $(2^{-78} \cdot 2^{82})^2 \cdot 16 = 2^{12}$. On the other hands, the probability that a fixed two-bit difference in Case 1 is connected with the $P_{2/1}$ of V'_{1_1} or V'_{2_7} is $2^{-10} \cdot 2 = 2^{-9}$ and thus the expected number of hits for each wrong 3-bit subkey is about $(2^{-72-9} \cdot 2^{82})^2 \cdot 16 \cdot 2^{-3} = 2^3$. This argument can be also applied to Cases 2 and 3. In Case 2, the expected number of hits for each right 4-bit subkey is about $(2^{-72-8} \cdot 2^{82})^2 \cdot 16 = 2^8$ and in Case 3, the expected number of hits for each right 5-bit subkey is about $(2^{-72-10} \cdot 2^{82})^2 \cdot 16 = 2^4$. Indeed, during the above procedure, subkey candidates can be checked by the overlapped control vectors (this fact makes easy to find the right subkey material). Taking into account these overlapped values, we retrieve 21 bits of the key by this attack. This attack can also retrieve the whole of master key pair (K, K') by performing an exhaustive search for the remaining keys and thus we can find the master key pair with a data complexity of 2^{83} related-key chosen plaintexts and a time complexity of 2^{107} encryptions.

5.2 Attack Procedure on Cobra-F64

Using two attack algorithms presented in previous subsection, we can similarly devise key recovery attacks on 11-round Cobra-F64a and 18-round Cobra-F64b. As for 11 rounds of Cobra-F64a, the above second attack scenario can be efficiently applied with some modifications, and as for 18 rounds of Cobra-F64b, the above first attack scenario can be efficiently applied with some modifications.

Let us first consider 11-round Cobra-F64a. In the second attack scenario, Step 2 collects ciphertext pairs whose differences satisfy (e_1, t') where plaintext pairs have the (e_1, e_1) difference and the master key pair has the $(0, 0, 0, e_1)$ difference. In Steps 3 and 4, for each 32-bit subkey K_2 we check the number of ciphertext pairs satisfying $C_i + K_2, C'_i + K_2 \in S(V)$, where $S(V)$ is a set of all 32-bit control vectors such that the control bits extracted by ciphertext pairs (C_i, C'_i) in Cases 1, 2, or 3 are fixed. In this way, we find out a group of 32-bit subkeys K_2 with maximal number of hits. With this group, we do an exhaustive search for the remaining 96-bit keys.

In this attack, we use a 11-round related-key differential characteristic $(e_1, e_1) \to (e_1, t')$ of Cobra-F64a which includes the FT. This characteristic can be derived from Table 6 by cutting off the 12-th round. The probability that t' is in Case 1 whose two BOP's are specified is about $2^{-48} \cdot 2^{-6} = 2^{-54}$, and the probability that t' is in Case 2 (resp., Case 3) whose two BOP's are specified is about $2^{-48} \cdot 2^{-8} = 2^{-56}$ (resp., $2^{-48} \cdot 2^{-10} = 2^{-58}$). Thus, if we use 2^{58} plaintext pairs, the expected number of hits for the right subkey K_2 follows the number of summing over the three expectations of Cobra-S128, i.e., $2^{12} + 2^8 + 2^4$, but the expected number of hits for a wrong subkey is much less than this value. Since the total number of control bits extracted in this attack is 21, the expected number of subkeys in the group is 2^{11}. Hence we can retrieve the whole of master key pair with a data complexity of 2^{59} related-key chosen plaintexts and a time complexity of 2^{107} encryptions.

We now consider 18-round Cobra-F64b. In the first attack scenario, we use a 18-round related-key differential characteristic $(0, e_1) \to (e_1 \oplus t^{\ggg 8}, t)$ of Cobra-F64b with probability 2^{-56} (which includes the FT). This characteristic can be derived from Table 7 by cutting off the last two rounds, i.e., rounds 19 and 20. Thus if we use 2^{57} desired plaintext pairs (the filtering rate in Step 2 is $2^{-64} \cdot 2^6 = 2^{-58}$), we can extract at least 6 bits of the control vector V by the same analysis as Cobra-S128. It follows that equation $V = C_L \oplus K_2 - (C_R \oplus K_3)^{\ggg 8}$ enables us to get 6 bits of key information, where C_L is the left 32 bits of ciphertext and C_R is the right 32 bits of ciphertext. Hence we can retrieve the whole of master key pair with a data complexity of 2^{58} related-key chosen plaintexts and a time complexity of $2^{128-6} = 2^{122}$ encryptions.

6 Conclusion

Three Cobra ciphers (Cobra-S128, Cobra-F64a, and Cobra-F64b) are considerably resistant against conventional attacks, e.g., the differential attack, the linear attack, and so on because they use data-dependent permutaions which are composed of the basic CP-boxes, $P_{2/1}$ (bit-controlled transpositons of two input bits). However, the very simple key schedule (i.e., the part of secret key is directly used in each round) and low diffusion of CP-boxes allow us to devise the related-key differential attacks on three Cobra ciphers.

In this paper, we presented the related-key attacks on Cobra-S128, Cobra-F64a and, Cobra-F64b. In the case of Cobra-S128, we can successfully recover 128-bit master keys of full-round Cobra-S128 with 2^{83} related-key chosen plaintexts and 2^{107} encryption units. In the cases of Cobra-F64a and Cobra-F64b, we can retrieve 128-bit master keys of 11-round Cobra-F64a and 18-round Cobra-F64b using 2^{59} and 2^{58} related-key chosen plaintexts, and 2^{107} and 2^{122} encryption units, respectively.

Acknowledgments

We would like to thank an anonymous reviewer for useful and interesting comments about this work.

References

1. E. Biham and A. Shamir, “Differential Cryptanalysis of the Data Encryption Standard”, Springer-Verlag, 1993.
2. N. D. Goots, B. V. Izotov, A. A. Moldovyan, and N. A. Moldovyan, “Modern cryptography: Protect Your Data with Fast Block Ciphers”, Wayne, A-LIST Publish., 2003.
3. N. D. Goots, B. V. Izotov, A. A. Moldovyan, and N. A. Moldovyan, “Fast Ciphers for Cheap Hardware : Differential Analysis of SPECTR-H64”, *MMM-ACNS’03*, LNCS 2776, Springer-Verlag, 2003, pp. 449-452.
4. N. D. Goots, N. A. Moldovyan, P. A. Moldovyanu and D. H. Summerville, “Fast DDP-Based Ciphers: From Hardware to Software”, *46th IEEE Midwest International Symposium on Circuits and Systems*, 2003.
5. N. D. Goots, A. A. Moldovyan, N. A. Moldovyan, “Fast Encryption ALgorithm Spectr-H64”, *MMM-ACNS’01*, LNCS 2052, Springer-Verlag, 2001, pp. 275-286.
6. S. Kavut and M. D. Yücel, “Slide Attack on Spectr-H64”, *INDOCRYPT’02*, LNCS 2551, Springer-Verlag, 2002, pp. 34-47.
7. J. Kelsey, B. Schneier, and D. Wagner, “Key Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES”, *Advances in Cryptology - CRYPTO ’96*, LNCS 1109, Springer-Verlag, 1996, pp. 237-251.
8. J. Kelsey, B. Schneier, and D. Wagner, “Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA”, *ICICS’97*, LNCS 1334, Springer-Verlag, 1997, pp. 233-246.

9. J. Kim, G. Kim, S. Hong, S. Lee and D. Hong, "The Related-Key Rectangle Attack - Application to SHACAL-1", *ACISP 2004*, LNCS 3108, Springer-Verlag, 2004, pp. 123-136.
10. J. Kim, G. Kim, S. Lee, J. Lim and J. Song, "Related-Key Attacks on Reduced Rounds of SHACAL-2", *INDOCRYPT 2004*, LNCS 3348, Springer-Verlag, 2004, pp. 175-190.
11. Y. Ko, D. Hong, S. Hong, S. Lee, and J. Lim, "Linear Cryptanalysis on SPECTR-H64 with Higher Order Differential Property", *MMM-ACNS03*, LNCS 2776, Springer-Verlag, 2003, pp. 298-307.
12. Y. Ko, C. Lee, S. Hong and S. Lee, "Related Key Differential Cryptanalysis of Full-Round SPECTR-H64 and CIKS-1 ", *ACISP 2004*, LNCS 3108, 2004, pp. 137-148.
13. Y. Ko, C. Lee, S. Hong, J. Sung and S. Lee, "Related-Key Attacks on DDP based Ciphers: CIKS-128 and CIKS-128H ", *Indocrypt 2004*, LNCS 3348, Springer-Verlag, 2004, pp. 191-205.
14. C. Lee, D. Hong, S. Lee, S. Lee, H. Yang, and J. Lim, "A Chosen Plaintext Linear Attack on Block Cipher CIKS-1", *ICICS 2002*, LNCS 2513, Springer-Verlag, 2002, pp. 456-468.
15. M. Matsui, "Linear cryptanalysis method for DES cipher", *Advances in Cryptology - EUROCRYPTO'93*, LNCS 765, Springer-Verlag, 1993, pp. 386-397.
16. A. A. Moldovyan and N. A. Moldovyan, "A cipher Based on Data-Dependent Permutations", *Journal of Cryptology*, volume 15, no. 1 (2002), pp. 61-72
17. R. C.-W Phan and H. Handschuh, "On Related-Key and Collision Attacks: The case for the IBM 4758 Cryptoprocessor", *ISC 2004*, LNCS 3225, Springer-Verlag, 2004, pp. 111-122.

A The Round Function *Crypt*$^{(e)}$ Used in Cobra-S128 and Cobra-F64

Fig. 6 represents the round function of Cobra-S128 and Fig. 7 represents the round function of Cobra-S64.

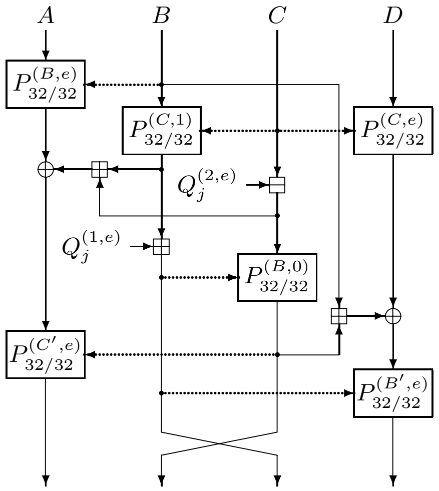


Fig. 6. Round function *Crypt*$^{(e)}$ of Cobra-S128

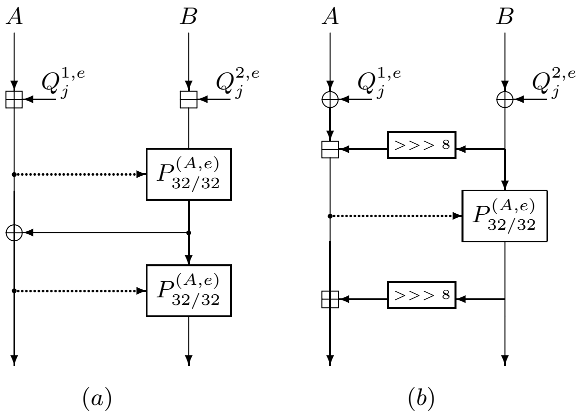


Fig. 7. (a) *Crypt*$^{(e)}$ of Cobra-F64a, (b) *Crypt*$^{(e)}$ of Cobra-F64b

B Related-Key Differential Characteristics of Cobra-F64a and Cobra-F64b

Note that the probability 2^{-2} in the FT line of Table 6 is derived from the last $\boxplus$ operation. Also note that the probability 2^{-5} in the line of the 20*th* round of Table 7 is derived from $P1$ and the last $\boxplus$ operation.

Table 6. Related-Key Differential Characteristic of Cobra-F64a

Round (i)	ΔRI^i	ΔRK^i	$P1/P2$	Prob.
1	(e_1, e_1)	$(0, e_1)$	$2^{-3}/2^{-3}$	2^{-6}
2	$(0, e_1)$	$(0, 0)$	$2^{-6}/2^{-8}$	2^{-14}
3	(e_1, e_1)	$(0, 0)$	$2^{-8}/2^{-6}$	2^{-14}
4	$(e_1, 0)$	$(e_1, 0)$	$1/1$	1
5	$(0, 0)$	$(0, 0)$	$1/1$	1
6	$(0, 0)$	$(0, 0)$	$1/1$	1
7	$(0, 0)$	$(e_1, 0)$	$2^{-3}/2^{-3}$	2^{-6}
8	$(0, e_1)$	$(0, e_1)$	$1/1$	1
9	$(0, 0)$	$(0, 0)$	$1/1$	1
10	$(0, 0)$	$(0, 0)$	$1/1$	1
11	$(0, 0)$	$(e_1, 0)$	$2^{-3}/2^{-3}$	2^{-6}
12	$(0, e_1)$	$(0, 0)$	$2^{-6}/2^{-8}$	2^{-14}
FT	(e_1, t)	$(0, 0)$	·	2^{-2}
Output	(e_1, t)	·	·	·
Total	·	·	·	2^{-62}

Table 7. Related-Key Differential Characteristic of Cobra-F64b

Round (i)	ΔRI^i	ΔRK^i	$P1$	Prob.
1	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-3}
2	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-3}
3	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-3}
4	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-3}
5	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-3}
⋮	⋮	⋮	⋮	⋮
17	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-3}
18	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-3}
19	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-3}
20	$(0, e_1)$	(e_1, e_1)	2^{-3}	2^{-5}
FT	$(e_1 \oplus t^{\ggg 8}, t)$	(e_1, e_1)	·	·
Output	$(e_1 \oplus t^{\ggg 8}, t)$	·	·	·
Total	·	·	·	2^{-62}

Advanced Slide Attacks Revisited: Realigning Slide on DES

Raphael C.-W. Phan

Information Security Research (iSECURES) Lab,
Swinburne University of Technology (Sarawak Campus),
93576 Kuching, Sarawak, Malaysia
rphan@swinburne.edu.my

Abstract. Slide attacks are powerful tools that enable the cryptanalyst to break ciphers with up to 4-round self-similarity. This paper introduces an advanced sliding technique that breaks ciphers with self-similarity more than 4 rounds, and even allows for sliding encryptions with dissimilar rounds in the middle of the slide. In particular, we present the *realigning slide attack* on variants of 14-, 15- and full 16-round DES. We hope our results will spur more effort into ways to extend the slide attacks to apply to larger classes of block ciphers with complex key schedules.

1 Introduction

The *slide attack* was introduced by Biryukov and Wagner in 1999 as a means to attack block ciphers by exploiting slight weaknesses in their key schedules [5]. A year later, the same authors presented advanced slide attacks [6], namely the *complementation slide* and *sliding with a twist* that could attack ciphers with slightly more complex key schedules. Now that 5 years have passed since then, it is natural to wonder if there are other ways to extend these slide attacks.

The DES' linear key schedule surprisingly resists all previously known slide and related-key [2,11,12] attacks, thus it is of major interest to show how it can be susceptible to slide attacks. Note however that linearity itself does not automatically imply weakness against such attacks. However, linear key schedules mean that relationships between round keys are much simpler both to exploit and possibly control, thus may have a higher chance of causing self-similarities. But DES has so far proven this wrong.

We introduce an advanced slide attack, the *realigning slide attack* by using a novel sliding technique that allows for sliding encryptions with dissimilar rounds in middle of the slide. Previously known sliding techniques would fail under this circumstance. We illustrate this new approach on DES variants, including the full 16 rounds with the original key schedule for a fraction of all keys, and slightly tweaked key schedules for almost all keys. Although our attack on full DES has a higher complexity than the best known attack, i.e. standard linear cryptanalysis [17] and is more of theoretical interest, our results indicate the irregular structure of the DES key schedule still has some exploitable degree of self-similarity that is susceptible to more subtle forms of slide attacks.

This paper is organized as follows: In Section 2, we briefly describe conventional and previous advanced slide attacks. We develop an advanced sliding technique in Section 3, the realigning slide which is demonstrated on DES variants. We discuss some related work in Section 4. We conclude in Section 5 and outline some interesting open problems in relation to extending the slide attacks.

2 The Slide Attacks

The basic slide attack [5] considers ciphers where each round is identical to the other. The cryptanalyst is interested to find a plaintext pair, P, P' with corresponding ciphertexts, C, C' such that he gets two *slid equations* of the form:

$$P' = F(P) \tag{1}$$

$$C' = F(C), \tag{2}$$

where $F(\cdot)$ is the round function. To do so, he obtains a pool of $2^{n/2}$ known plaintexts (KPs) and corresponding ciphertexts (n is the block size), and uses this to form 2^n pairs. He then either directly checks if each pair satisfies the slid equations or has to make guesses of the keys in F while doing the checking. By the birthday paradox, he expects one slid pair satisfying the equations, upon which the key used in the F is recovered.

The limitation of this conventional technique is that it applies only to a small class of ciphers, particularly those whose key schedules cause identical round keys for each round, thus making each round identical to the other. This technique is basically a clever adaptation of the rotating subkey related-key attack[1] [2] to the non-related-key context, i.e. the requirement for related keys that cause identical or self-similar (repeating) round keys is eliminated by using ciphers with weak key schedules that themselves cause identical or self-similar round keys.

2.1 Advanced Sliding Techniques

The basic slide attack works on one-round self-similar ciphers, i.e. all round keys are identical, but when the self-similarity consists of more complex rounds, then further advanced sliding techniques have to be used. Two such techniques: complementation slide and twisting slide, were presented in [6].

The *complementation slide* applies particularly well to Feistel-like ciphers and amplifies their two-round self-similarity into one-round self-similarity. The basic concept is to slide two encryptions such that the slid rounds, rather than being exactly identical to each other, have a constant difference due to dissimilar round keys that propagates with probability one from one end of the slid rounds to the other. In this way, the plaintexts and ciphertexts forming a slid pair are still related by one unslid round and the slid equations are similar to (1) and (2). The restriction is that the round keys must be inserted via the same

[1] In modern day terms, this is more suitably known as related-key slide attack.

operation as that used to combine the two Feistel halves, and that there be no nonlinear operation between the round key insertion operation and the Feistel half combining operation. For more details of why this is so, see [6,3].

The *sliding with a twist* (*twisting slide*) technique slides an encryption with a decryption and is equally applicable to ciphers with two-round self-similarity, and even much complex ciphers such as DESX proposed by Rivest [13,14].

The two above-mentioned techniques can also be combined into the *complementation sliding with a twist* to attack ciphers with up to four-round self-similarity. But this has restrictions similar to the complementation slide and only applies to Feistel-like ciphers.

The basic and advanced slide attacks are powerful tools in that they are applicable to ciphers independent of the number of rounds. Other slide-style attacks are in [19,7,10,20]. However, current slide attacks are only applicable to cipher key schedules with self-similarity up to 4 rounds, and if there are no unslid rounds in the middle of a slid sequence. In the next section, we show how the slide attacks can be made to overcome these two limitations.

3 The Realigning Slide Attack

We introduce a new advanced sliding technique: the *realigning slide*, and show how it slides in a way that previous sliding techniques were unable to.

We describe how to slide two encryptions even with unslid rounds in the middle of the slide. In essence, the technique allows the encryptions to realign themselves with each other even after a misalignment caused by the middle unslid rounds. To illustrate this concept, we apply it to the DES key schedule with up to its full 16 rounds, but we emphasize that these attacks presented here apply to ciphers with DES-like key schedules with an infinite number of rounds.

DES is a Feistel cipher with 64-bit blocks and a 56-bit key. The 64-bit plaintext block goes through a 16-round Feistel structure to obtain the ciphertext[2]. Denote the plaintext block as two halves, $L_0||R_0$, where $||$ denotes concatenation; then iterate 15 rounds (for $i = 1, 2, \ldots, 15$):

$$L_i \leftarrow R_{i-1} \tag{3}$$

$$R_i \leftarrow L_{i-1} \oplus F(R_{i-1}, K_i), \tag{4}$$

where $F(\cdot, K_i)$ denotes the F function keyed by round key K_i. Finally, do:

$$L_{16} \leftarrow L_{15} \oplus F(R_{15}, K_{16}) \tag{5}$$

$$R_{16} \leftarrow R_{15}, \tag{6}$$

i.e. there is no swap in the final (16th) round.

The key schedule of DES takes a 64-bit secret key which is passed through a permutation, $PC1$ that removes the parity bits, causing a resultant 56-bit key,

[2] Our DES description here is kept concise, and neglects details that do not contribute to its security, i.e. initial and final permutations, etc. For these and further details of DES, we refer the reader to [18].

K. Since this permutation is of no cryptographic importance, the secret key of DES is normally assumed to be 56 bits in length. The 56-bit key, K is divided into two halves, C_0 and D_0, each of 28 bits, hence we have $K = C_0||D_0$. The round keys, K_i where $i \in \{1, \ldots, 16\}$ are defined as $K_i = PC2(C_i||D_i)$ where $C_i = LS_i(C_{i-1}), D_i = LS_i(D_{i-1})$, $PC2$ is a permutation and where LS_i is a left circular shift by the number of positions according to Table 1.

Table 1. Circular shifts in the key schedule of DES

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LS_i	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
$a[i]$	1	2	4	6	8	10	12	14	15	17	19	21	23	25	27	28

In this paper, we will use the alternative representation introduced by Knudsen in [15,16], and define $L_{a[i]}(C_0||D_0) = LS_{a[i]}(C_0)||LS_{a[i]}(D_0)$ where $a[i]$ is the accumulated number of shifts given in Table 1. Hence, the round keys are then $K_i = PC2(L_{a[i]}(C_0||D_0)) = PC2(L_{a[i]}(K))$.

The DES key schedule has undergone years of analysis, and despite being linear, has remained strong against attacks that exploit key schedules such as related-key attacks and slide attacks. Thus, it has defied the popular belief among cryptographers that strong key schedules should be nonlinear.

An attempt to attack the DES key schedule with related-key cryptanalysis was made by Biham in 1994 [2], but was only applicable to a very much weakened version where the number of shifts in each round is constant (and regular). This version is really not in line with the original DES design: that of having irregularity in the key schedule. Also, past research has shown that key schedules with some form of regularity [2,5,6] fall to related-key and slide attacks.

In this section, we show how to slide the original unmodified DES key schedule despite its irregularity. For this purpose, we exploit results by Knudsen [15]. Denote the secret key as K and the i^{th} round key generated from K as K_i. Then

***Theorem 1* (Knudsen [15]).** For every key K, there exists a key K', s.t.

$$K_{i+1} = K'_i; \qquad i \in \{2, \ldots, 7\} \cup \{9, \ldots, 14\}$$

i.e. K and K' have 12 common round keys.

***Theorem 2* (Knudsen [15]).** There exist 2^8 pairs of keys K and K', s.t.

$$K_{i+1} = K'_i; \qquad i \in \{2, \ldots, 14\}$$

i.e. K and K' have 13 common round keys.

We first exploit Theorem 1 here to motivate the intuition behind the realigning slide. Theorem 2 will be used later in Section 3.1 to demonstrate (as an initial step towards developing the realigning slide,) a conventional related-key slide attack on a variant of the DES for the class of 2^8 key-pairs of Theorem 2.

Consider two encryptions keyed by K and K' respectively (of Theorem 1). Then the rounds 3 to 8, 10 to 15 of the first encryption (we denote round i by r_i) share common round keys with rounds 2 to 7, 9 to 14 of the second encryption (denote round i by r'_i). This naively suggests a possible slide of the encryptions such that they are out of phase by 1 round, denoted below, where bold round numbers indicate the slid rounds:

$$\begin{array}{l}
\qquad\qquad X \qquad\qquad\qquad\qquad\quad Y \qquad\qquad\qquad\qquad\quad Z \\
P \rightarrow 1 \;\; 2 \;\; \mathbf{3} \;\; \mathbf{4} \;\; \mathbf{5} \;\; \mathbf{6} \;\; \mathbf{7} \;\; \mathbf{8} \;\; 9 \;\; \mathbf{10} \;\; \mathbf{11} \;\; \mathbf{12} \;\; \mathbf{13} \;\; \mathbf{14} \;\; \mathbf{15} \;\; 16 \rightarrow C \\
\;\; P' \rightarrow 1 \;\; \mathbf{2} \;\; \mathbf{3} \;\; \mathbf{4} \;\; \mathbf{5} \;\; \mathbf{6} \;\; \mathbf{7} \;\; 8 \;\; \mathbf{9} \;\; \mathbf{10} \;\; \mathbf{11} \;\; \mathbf{12} \;\; \mathbf{13} \;\; \mathbf{14} \;\; 15 \;\; 16 \rightarrow C' \\
\qquad\qquad X' \qquad\qquad\qquad\qquad\quad Y' \qquad\qquad\qquad\qquad\quad Z'
\end{array}$$

The first slid sequence is ($r_3 \sim r_8$, $r'_2 \sim r'_7$), while the second slid sequence is ($r_{10} \sim r_{15}$, $r'_9 \sim r'_{14}$). Nevertheless, the conventional and advanced slide attacks are not applicable in this case since we have an unslid round in the middle of the slide, namely in (r_9, r'_8). Due to this unslid round, even though the first slid sequence has been aligned properly, a misalignment occurs and hence impedes the second slid sequence from being aligned as well.

We propose to force the slide of the second sequence probabilistically. Denote the outputs of the middle unslid round (r_9, r'_8) by Y and Y'; the outputs of (r_2, r'_1) by X and X', and the outputs of (r_{15}, r'_{14}) by Z and Z', respectively. The intuition is that once we have a pair such that $X = X'$, we have $Y = Y'$ with a certain probability p, and hence Z = Z' is obtained for free.

Such a pair is a slid pair that satisfies the following slid equations:

$$E_{1,2}(P) = E_1(P') \tag{7}$$

$$E_{16}^{-1}(C) = E_{15,16}^{-1}(C') \tag{8}$$

where $E_{i_1,i_2,...,i_x}$ (respectively $E_{i_1,i_2,...,i_x}^{-1}$) denotes x rounds of the encryption (respectively decryption) keyed by the corresponding round keys of the rounds $i_1, i_2, ..., i_x$. From these slid equations, we extract the relations[3]:

$$P'_R = P_L \oplus F(P_R, K_1) \tag{9}$$

$$C'_L = C_R \oplus F(C'_R, K'_{16}) \tag{10}$$

where $P = P_L || P_R$, etc.

Essentially, we have realigned the misalignment caused by the middle unslid round, and we call this the *realigning slide.* This technique allows for sliding encryptions with unslid middle rounds.

The next question: What is the probability, p such that $Y' = Y$? At the input to (r_9, r'_8), the text inputs are equal due to the sliding condition of the first sequence, but the round keys (K_9, K'_8) are different, so feeding in the key difference to the round function, what is the chance to have a zero output difference?

[3] Recall there is no swap in the final round of DES.

For this, we recall the round function of DES. In each round, the 32-bit text input is expanded to 48 bits before being XORed to a 48-bit round key. This result is then passed through 8 Sboxes to produce a 32-bit output which is then put through a bit permutation. This forms the round function output. In our case, the 32-bit text input from both encryptions is the same, and hence their difference cancels out to zero. Thus, the input difference to the Sboxes comes only from the round key difference. Referring to the difference distribution table (DDT) [1] for each Sbox, we note that out of 64 possible input differences, around 34 to 38 (except for Sbox 4 which has 25) may cause a zero output difference, so with probability on average $34.25/64 \approx 2^{-1}$ for each Sbox we get an input difference that may cause a zero output difference, and thus 2^{-8} for all 8 Sboxes. This means out of all possible key pairs (K, K') of Theorem 1, we have roughly $2^{56} \times 2^{-8} = 2^{48}$ pairs [15] whose round keys input to (r_9, r'_8) would have an input difference that may cause a zero output difference and thus $Y' = Y$.

Based on this discussion, we outline attacks on variants of DES that use the original DES key schedule, or with slight tweaks to 1 or 2 rounds but where the irregular structure is still preserved.

3.1 Sliding the Middle 14 DES Rounds

Consider the middle 14 rounds of the DES, namely from round 2 to round 15, using the original key schedule [4], namely with the shift pattern 12222221222222. Then this variant is susceptible to the realigning slide as outlined above, where sliding by one round we obtain

$$\begin{array}{l} P \rightarrow 2\ \ \mathbf{3}\ \ \mathbf{4}\ \ \mathbf{5}\ \ \mathbf{6}\ \ \mathbf{7}\ \ \mathbf{8}\ \ 9\ \ \mathbf{10}\ \ \mathbf{11}\ \ \mathbf{12}\ \ \mathbf{13}\ \ \mathbf{14}\ \ \mathbf{15} \rightarrow C \\ \quad P' \rightarrow \mathbf{2}\ \ \mathbf{3}\ \ \mathbf{4}\ \ \mathbf{5}\ \ \mathbf{6}\ \ \mathbf{7}\ \ 8\ \ \mathbf{9}\ \ \mathbf{10}\ \ \mathbf{11}\ \ \mathbf{12}\ \ \mathbf{13}\ \ \mathbf{14}\ \ 15 \rightarrow C' \end{array}$$

Before we discuss this realigning slide, we first consider a class of 2^8 key-pairs (Theorem 2) such that their round keys input to the (r_9, r'_8) rounds are the same thus no middle unslid round, and hence we can attack this middle 14 rounds of the DES with a conventional related-key slide attack. We will then explain how to convert this into a realigning slide that applies for all DES keys.

Conventional Related-key Slide. For the class of 2^8 key-pairs of Theorem 2, we have that the two encryption sequences:

$$\begin{array}{l} P \rightarrow 2\ \ \mathbf{3}\ \ \mathbf{4}\ \ \mathbf{5}\ \ \mathbf{6}\ \ \mathbf{7}\ \ \mathbf{8}\ \ \mathbf{9}\ \ \mathbf{10}\ \ \mathbf{11}\ \ \mathbf{12}\ \ \mathbf{13}\ \ \mathbf{14}\ \ \mathbf{15} \rightarrow C \\ \quad P' \rightarrow \mathbf{2}\ \ \mathbf{3}\ \ \mathbf{4}\ \ \mathbf{5}\ \ \mathbf{6}\ \ \mathbf{7}\ \ \mathbf{8}\ \ \mathbf{9}\ \ \mathbf{10}\ \ \mathbf{11}\ \ \mathbf{12}\ \ \mathbf{13}\ \ \mathbf{14}\ \ 15 \rightarrow C' \end{array}$$

In more detail, this class of key-pairs is where we have restrictions (fixed values) on 48 bits of K to be equal to 48 other bits of K' such that $K_9 = K'_8$.

So the two sequences will be slid on the rounds $(r_3 \sim r_{15},\ r'_2 \sim r'_{14})$ and no middle unslid round, thus we have the slid equations:

$$P' = r_2(P) \tag{11}$$

[4] Note that the original shift pattern for the full 16 rounds is 1122222212222221.

$$C' = r'_{15}(C). \tag{12}$$

Since DES has a Feistel structure, the slid equations can be simplified to:

$$P'_L = P_R \tag{13}$$

$$P'_R = P_L \oplus F(P_R, K_2). \tag{14}$$

$$C'_R = C_L \tag{15}$$

$$C'_L = C_R \oplus F(C'_R, K'_{15}). \tag{16}$$

Equation (13) can be exploited in forming chosen plaintext queries with much less text complexity, namely we form 2^{16} chosen plaintexts P and another 2^{16} chosen plaintexts P' such that $P'_L = P_R$, and obtain their encryptions keyed by K and K', respectively. These form 2^{32} pairs P, P' from which we expect to get 1 slid pair such that $X = X'$ and thus $Z = Z'$.

Out of these 2^{32} pairs, we have a 32-bit filtering condition (equation 15) on the ciphertexts namely $C'_R = C_L$ so from the total 2^{32} pairs only one or two pairs would pass through to the next phase of analysis. For these few remaining pairs, equations (14,16) are each used to obtain 2^{16} possible values[5] of each for 48-bit K_2 and 48-bit K'_{15}, or in total 2^{32} values for (K_2, K'_{15}). Since 40 bits[6] of K_2 and K'_{15} are in common, each pair on average suggests $2^{32} \times 2^{-40} = 2^{-8}$ candidates, so only the right value of (K_2, K'_{15}) remains.

In all, this requires 2^{17} related-key chosen plaintexts (*RK-CP*s) and negligible effort since processing each remaining slid pair on equations (14,16) is equivalent to just 2 DES rounds.

Realigning Slide. We extend this to apply for almost all key-pairs (2^{48}). For any key, K of the DES, and a corresponding related key, K' (of Theorem 1), the two encryption sequences keyed by them respectively would be slid on round sequences ($r_3 \sim r_8$, $r'_2 \sim r'_7$) and ($r_{10} \sim r_{15}$, $r'_9 \sim r'_{14}$), thus we have an unslid round (r_9, r'_8) in the middle, causing our previous related-key slide attack to fail.

However, recall from our discussion before Section 3.1 that there are 2^{48} pairs of keys such that a non-zero input difference to (r_9, r'_8) may cause a zero output difference and thus $Y = Y'$.

In the general case, if the probability of getting a zero output difference of (r_9, r'_8) is p, then obtain the encryptions of $\sqrt{(1/p)} \times 2^{16}$ chosen plaintexts P and another $\sqrt{(1/p)} \times 2^{16}$ chosen plaintexts P', keyed by K and K' respectively such that $P'_L = P_R$. These form $(1/p) \times 2^{32}$ pairs, (P, P') from which we expect to get $1/p$ potential slid pairs such that $X = X'$. Out of these, we further expect with a probability p that $Y = Y'$ and hence $Z = Z'$, so one slid pair exists which satisfies the slid equations (11,12).

[5] Consider equation (14). Work inwards with known P'_R, P_L, P_R values, including traversing each of 8 DES Sboxes in reverse which suggests 2^2 inputs until we can calculate the bits of K_2, thus 2^2 possible values for each 6-bit subset of K_2 that influences each Sbox input, so $2^{2\times 8} = 2^{16}$ in total. Similarly for equation (16) for K'_{15}.

[6] This has been experimentally verified by a C program on the DES key schedule.

From equation (15) we have a 32-bit filtering condition on the ciphertexts namely $C'_R = C_L$, so out of the total $(1/p) \times 2^{32}$ pairs only $1/p$ pairs would pass through to the next phase of analysis. Each remaining pair suggests 2^{16} candidates for K_2 via slid equation (14) and another 2^{16} candidates for K'_{15} via slid equation (16), thus a total of 2^{32} for (K_2, K'_{15}). Since they are common in 40 bits, on average each pair suggests $2^{32} \times 2^{-40} = 2^{-8}$ candidates. As there are $1/p$ such pairs, the total number of candidates is $1/p \times 2^{-8}$.

This gives a realigning slide on the middle 14 rounds of DES that applies for almost all keys, requiring $\sqrt{(1/p)} \times 2^{17}$ related-key chosen plaintexts. Since checking each pair is equivalent to 2 one-round DES encryptions, the time required by the analysis phase is $1/p \times 1/7$ DES encryptions.

We discuss as a concrete example how this works for any related key-pair, K and K' having round keys to (r_9, r'_8) with difference `03 3B 22 2E 26 22 2B 28` (each difference is arranged in 8 groups of 6 bits each, in hex) as input to the 8 Sboxes. Each 6-bit difference has the highest probability[7] of causing a zero output difference for its corresponding Sbox, namely the resultant probability, p of $14/64 \times 16/64 \times 12/64 \times 16/64 \times 14/64 \times 16/64 \times 16/64 \times 10/64 \approx 2^{32.5}/2^{48} = 2^{-15.5}$ (These are trivially obtained from the DDT of DES' Sboxes [1]).

Thus, for such a key-pair, a realigning slide on the middle 14 DES rounds requires $\sqrt{(1/p)} \times 2^{17} = \sqrt{2^{15.5}} \times 2^{17} = 2^{24.75}$ related-key chosen plaintexts and time of $1/p \times 1/7 = 2^{15.5} \times 1/7 \approx 2^{12.5}$ DES encryptions in the best case.

Besides key-pairs of this difference, there are other similar key-pairs that cause a zero output difference after (r_9, r'_8) with the same probability p, including those with the round key differences to (r_9, r'_8) as (`03 3B 22 2E 27 22 2B 28`), (`03 3B 22 2F 26 22 2B 28`) and (`03 3B 22 2F 27 22 2B 28`). These are just a few examples out of the 2^{48} key-pairs of DES that cause a zero output difference after (r_9, r'_8) with probability p which ranges from $2^{-15.5}$ to $2/64 \times 2/64 \times 2/64 \times 4/64 \times 2/64 \times 2/64 \times 2/64 \times 2/64 = 2^9/2^{48} = 2^{-39}$. If we did not restrict on a specific round key difference to (r_9, r'_8), then p is 2^{-28} by averaging for each Sbox over all non-zero input differences that cause a zero output difference, thus the text and attack complexities from the best to average case would increase by a factor of $2^{6.25}$ and $2^{12.5}$, respectively.

3.2 The First 15 DES Rounds

We now consider two variants of the first 15 rounds of the DES, namely with a slightly tweaked key schedule, and with the original key schedule.

Attack 1: Slightly Tweaked Key Schedule (Tweak in Round 2). The realigning attack in Section 3.1 equally applies to the DES-variant reduced to the first 15 rounds with a slightly tweaked key schedule, where only the shift in round 2 is modified to 2 instead of 1, i.e. the shift pattern becomes 1222222212222222 as compared to 1122222212222222 originally. Sliding by one round:

[7] The relation in these bits between the related key-pair can be chosen to this difference to maximize its probability.

$$\begin{array}{l}
P \rightarrow 1\ \mathbf{2}\ \mathbf{3}\ \mathbf{4}\ \mathbf{5}\ \mathbf{6}\ \mathbf{7}\ \mathbf{8}\ 9\ \mathbf{10}\ \mathbf{11}\ \mathbf{12}\ \mathbf{13}\ \mathbf{14}\ \mathbf{15} \rightarrow C \\
\quad P' \rightarrow \mathbf{1}\ \mathbf{2}\ \mathbf{3}\ \mathbf{4}\ \mathbf{5}\ \mathbf{6}\ \mathbf{7}\ 8\ \mathbf{9}\ \mathbf{10}\ \mathbf{11}\ \mathbf{12}\ \mathbf{13}\ \mathbf{14}\ 15 \rightarrow C'
\end{array}$$

hence we get a realigning slide with the same complexity as in Section 3.1.

Attack 2: Original Key Schedule for 2^8 Key-Pairs. Is the attack equally applicable to the first 15 rounds of DES using the original key schedule? Our answer is yes, for a certain class of DES key-pairs. At first glance, the analysis phase is harder due to the absence of the slid round (r_2, r'_1). Nevertheless, for a class of 2^8 key-pairs (formalized in Theorem 3) such that the two round keys K_2 and K'_1 generated from K and K' are the same, then the attack works as described previously for the middle 14 DES rounds. Note though that this is only of theoretical interest as we are attacking the 2^8 weak key-pairs with effort more than going through this weak-key sub-space.

Theorem 3. There exist 2^8 pairs of keys K and K', s.t.

$$K_{i+1} = K'_i; \qquad i \in \{1, \ldots, 7\} \cup \{9, \ldots, 14\}$$

i.e. K and K' have 13 common round keys.
Proof: From Theorem 1, we have that there is a related key, K' for every DES key, K such that they have 12 rounds in common. When we restrict on the 48 bits of K_2 to be identical to K'_1, we have 8 bits of freedom and hence a class of 2^8 key-pairs K and K' such that in addition to the 12 common rounds, an additional (13th) round (r_2, r'_1) is also in common. □

3.3 The Last 15 DES Rounds

Attack 1: Slightly Tweaked Key Schedule (Tweak in Round 16). Suppose we only tweak the shift in round 16, in that we modify the shift value from 1 to 2, we get the shift pattern 122222212222222. Sliding the two encryptions:

$$\begin{array}{l}
P \rightarrow 2\ \mathbf{3}\ \mathbf{4}\ \mathbf{5}\ \mathbf{6}\ \mathbf{7}\ \mathbf{8}\ 9\ \mathbf{10}\ \mathbf{11}\ \mathbf{12}\ \mathbf{13}\ \mathbf{14}\ \mathbf{15}\ \mathbf{16} \rightarrow C \\
\quad P' \rightarrow \mathbf{2}\ \mathbf{3}\ \mathbf{4}\ \mathbf{5}\ \mathbf{6}\ \mathbf{7}\ 8\ \mathbf{9}\ \mathbf{10}\ \mathbf{11}\ \mathbf{12}\ \mathbf{13}\ \mathbf{14}\ \mathbf{15}\ 16 \rightarrow C'
\end{array}$$

and the realigning slide attack in Section 3.1 applies directly.

Attack 2: Original Key Schedule for 2^8 Key-Pairs. For the last 15 rounds with the original key schedule, we no longer have the round keys to (r_{16}, r'_{15}) being the same, hence this is not a slid round. However, for a class of 2^8 key-pairs (formalized in Theorem 4) such that the two round keys K_{16} and K'_{15} generated from K and K' are the same, the attack as in Section 3.1 can be applied. Again though, this is only of theoretical interest since the effort is greater than going through the weak-key sub-space.

Theorem 4. There exist 2^8 pairs of keys K and K', s.t.

$$K_{i+1} = K'_i; \qquad i \in \{2, \ldots, 7\} \cup \{9, \ldots, 15\}$$

i.e. K and K' have 13 common round keys.
Proof: From Theorem 1, we have that there is a related key, K' for every DES key, K such that they have 12 rounds in common. When we restrict on the 48 bits of K_{16} to be identical to K'_{15}, we have 8 bits of freedom and thus a class of 2^8 key-pairs K and K' such that in addition to the 12 common rounds, an additional (13th) round (r_{16}, r'_{15}) is also in common. $\square$

Attack 3: Original Key Schedule. The last 15 DES rounds with the original key schedule can be attacked with the realigning slide even for the case where we do not have equal round keys for the round (r_{16}, r'_{15}), thus this applies for almost all (recall this is 2^{48} out of 2^{56}) the key-pairs. In such a case, we have:

$$\begin{array}{l} P \rightarrow 2\;\; \mathbf{3}\;\; \mathbf{4}\;\; \mathbf{5}\;\; \mathbf{6}\;\; \mathbf{7}\;\; \mathbf{8}\;\; 9\;\; \mathbf{10}\;\; \mathbf{11}\;\; \mathbf{12}\;\; \mathbf{13}\;\; \mathbf{14}\;\; \mathbf{15}\;\; 16 \rightarrow C \\ \quad P' \rightarrow \mathbf{2}\;\; \mathbf{3}\;\; \mathbf{4}\;\; \mathbf{5}\;\; \mathbf{6}\;\; \mathbf{7}\;\; 8\;\; \mathbf{9}\;\; \mathbf{10}\;\; \mathbf{11}\;\; \mathbf{12}\;\; \mathbf{13}\;\; \mathbf{14}\;\; 15\;\; 16 \rightarrow C' \end{array}$$

i.e. there is no longer any filtering condition on the ciphertexts, so although the slid equations (13,14) still apply, slid equations equations (15,16) are replaced with slid equation (10). The plus side with this situation is that we are aiming to recover K_2 and K'_{16} which are the same! Thus we have a 48-bit filtering condition on the keys suggested by equations (14,10) instead of just 40 bits.

We check the slid equations (14,10) on all the $(1/p) \times 2^{32}$ pairs instead of just $1/p$ had there been a filtering condition. Each pair suggests $2^{32} \times 2^{-48} = 2^{-16}$ key candidates, thus with $(1/p) \times 2^{32}$ pairs, the total number of suggested key candidates is $(1/p) \times 2^{32} \times 2^{-16} = (1/p) \times 2^{16}$.

As a concrete example, for the related key-pair, K and K' with round keys to (r_9, r'_8) having the difference `03 3B 22 2E 26 22 2B 28` given in Section 3.1, the total number of suggested candidates for (K_2, K'_{16}) is $(1/p) \times 2^{16} = 2^{15.5} \times 2^{16} = 2^{31.5}$, reduced from the keyspace of 2^{48}.

This needs the same number of texts as before, but due to more pairs checked, the time complexity is $(1/p) \times 2^{32} \times 1/7$ DES encryptions, or considering the above example $2^{15.5} \times 2^{32} \times 1/7 \approx 2^{44.5}$ such encryptions in the best case.

3.4 The Full 16 DES Rounds

We can extend the attack further to several variants of the full 16 DES rounds, even with the original key schedule.

Attack 1: Tweaked Key Schedule (Tweak in Round 2 and Round 16). If we modify the shift in round 2 and round 16, from 1 to 2, we get the shift pattern 1222222212222222. We then have:

$$\begin{array}{l} P \rightarrow 1\;\; \mathbf{2}\;\; \mathbf{3}\;\; \mathbf{4}\;\; \mathbf{5}\;\; \mathbf{6}\;\; \mathbf{7}\;\; \mathbf{8}\;\; 9\;\; \mathbf{10}\;\; \mathbf{11}\;\; \mathbf{12}\;\; \mathbf{13}\;\; \mathbf{14}\;\; \mathbf{15}\;\; \mathbf{16} \rightarrow C \\ \quad P' \rightarrow \mathbf{1}\;\; \mathbf{2}\;\; \mathbf{3}\;\; \mathbf{4}\;\; \mathbf{5}\;\; \mathbf{6}\;\; \mathbf{7}\;\; 8\;\; \mathbf{9}\;\; \mathbf{10}\;\; \mathbf{11}\;\; \mathbf{12}\;\; \mathbf{13}\;\; \mathbf{14}\;\; \mathbf{15}\;\; 16 \rightarrow C' \end{array}$$

and the realigning slide attack in Section 3.1 applies directly.

Attack 2: Tweaked Key Schedule (Tweak in Round 16). If we tweak the key schedule even less, namely we only modify the shift in round 16 from 1 to 2, we get the shift pattern 1122222212222222. Sliding two such encryptions by one round, we get:

$$\begin{array}{llllllllllllllllll}
P \rightarrow 1 & 2 & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & 9 & \mathbf{10} & \mathbf{11} & \mathbf{12} & \mathbf{13} & \mathbf{14} & \mathbf{15} & \mathbf{16} \rightarrow C & \\
 & P' \rightarrow 1 & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & 8 & \mathbf{9} & \mathbf{10} & \mathbf{11} & \mathbf{12} & \mathbf{13} & \mathbf{14} & \mathbf{15} & 16 \rightarrow C'
\end{array}$$

This situation is similar to Attack 2 in Section 3.2, where for a class of 2^8 key-pairs of Theorem 3, we get $K_2 = K'_1$ and we can then apply the attack in Section 3.1. Nevertheless, the effort is greater than brute-forcing the weak-key sub-space.

Attack 3: Tweaked Key Schedule (Tweak in Round 2). Similarly if we only modify the shift in round 2 from 1 to 2, we get the shift pattern 1222222212222221, and thus:

$$\begin{array}{llllllllllllllllll}
P \rightarrow 1 & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & 9 & \mathbf{10} & \mathbf{11} & \mathbf{12} & \mathbf{13} & \mathbf{14} & \mathbf{15} & 16 \rightarrow C & \\
 & P' \rightarrow \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & 8 & \mathbf{9} & \mathbf{10} & \mathbf{11} & \mathbf{12} & \mathbf{13} & \mathbf{14} & 15 & 16 \rightarrow C'
\end{array}$$

This is similar to Attack 2 and Attack 3 in Section 3.3, namely it applies for a class of 2^8 key-pairs as per Theorem 4 (with the addition of an extra 14th round (r_2, r'_1)), and for almost all key-pairs of the DES, respectively.

Attack 4: Original Key Schedule for $\mathbf{2^8}$ Key-Pairs. Finally, we look at the full 16-round DES with its original key schedule, where the shift pattern is 1122222212222221. This gives:

$$\begin{array}{llllllllllllllllll}
P \rightarrow 1 & 2 & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & 9 & \mathbf{10} & \mathbf{11} & \mathbf{12} & \mathbf{13} & \mathbf{14} & \mathbf{15} & 16 \rightarrow C & \\
 & P' \rightarrow 1 & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & 8 & \mathbf{9} & \mathbf{10} & \mathbf{11} & \mathbf{12} & \mathbf{13} & \mathbf{14} & 15 & 16 \rightarrow C'
\end{array}$$

Notice there is no filtering condition on the ciphertexts. For a class of 2^8 key-pairs as per Theorem 3, the situation becomes similar to Attack 3 in Section 3.3 (also with no filtering condition), thus this can be attacked with similar texts and time complexity. Fortunately for the full DES, the effort is more than brute-forcing the weak-key sub-space.

4 Discussion and Related Work

Wagner [22] has independently presented concepts similar to our realigning slide in the context of hash functions, i.e. in producing pseudo-collisions on 40 rounds of the SHA-1 hash function. SHA-1 has the Davies-Meyer (DM) construction:

$$H_i = E(M_i, H_{i-1}) \oplus H_{i-1}, \tag{17}$$

where $E(\cdot)$ is a compression round function, $M = M_1, \ldots, M_i, \ldots, M_n$ is the input message, H_i is the chaining variable, H_0 is a fixed initialization vector

(IV) and H_n is the hash output. Wagner proposes to overcome the misalignment (he calls them "defects in sliding") by collecting enough IVs and input texts so as to "bypass the defect" with high probability.

Saarinen [21] extended this to the full 80 rounds of SHA-1. It was also discussed how this may potentially be turned into a related-key attack on the SHACAL-1 block cipher [8,9], which is derived from SHA-1 by peeling off the DM final chaining (via $\oplus$), taking the IV as plaintext, the message blocks M_i as the round keys and thus the output H_n becomes the ciphertext. However, this uses a very restrictive model in that the cryptanalyst has to choose specific values for 16 consecutive round key words; thus destroying the motivation for key-recovery attacks.

5 Conclusion and Open Problems

We give in Table 2 a comparison of our best and average case realigning slide results on the DES with its original key schedule or with very minor tweaks in just 1 or 2 rounds. We also list Biham's [2] related-key slide attack on a very much weakened DES key schedule where all 16 rounds are tweaked such that they have the same shift amounts. Note that Biham's tweak destroys the original irregular structure of the DES key schedule, in contrast to the DES versions we attack where the irregular structure is preserved and thus better model the original DES design.

Though adapting this realigning slide to other cipher key schedules will need a detailed analysis of the specific individual key schedules, this is the first time that slide attacks can overcome unslid middle rounds, and that DES with its original key schedule is shown susceptible to slide attacks (thus highlighting that even irregular key schedules can be slid). Fortunately for the full 16-round DES with its original key schedule, this only applies for 2^8 key-pairs but we view

Table 2. Comparison of attacks on DES variants

Variant	Key Schedule	Number of Key-Pairs	Texts (Best/Average)	Encryptions (Best/Average)	Source
middle 14	Original	2^8	$2^{17} RK\text{-}CP$	-	Sec 3.1: Attack 1
middle 14	Original	2^{48}	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{12.5}/2^{25}$	Sec 3.1: Attack 2
first 15	Tweak round 2	2^{48}	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{12.5}/2^{25}$	Sec 3.2: Attack 1
first 15	Original	2^8	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{12.5}/2^{25}$	Sec 3.2: Attack 2
last 15	Tweak round 16	2^{48}	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{12.5}/2^{25}$	Sec 3.3: Attack 1
last 15	Original	2^8	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{12.5}/2^{25}$	Sec 3.3: Attack 2
last 15	Original	2^{48}	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{44.5}/\star$	Sec 3.3: Attack 3
full 16	Tweak round 2 & 16	2^{48}	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{12.5}/2^{25}$	Sec 3.4: Attack 1
full 16	Tweak round 16	2^8	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{12.5}/2^{25}$	Sec 3.4: Attack 2
full 16	Tweak round 2	2^8	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{12.5}/2^{25}$	Sec 3.4: Attack 3
full 16	Tweak round 2	2^{48}	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{44.5}/\star$	Sec 3.4: Attack 3
full 16	Original	2^8	$2^{24.75}/2^{31} RK\text{-}CP$	$2^{44.5}/\star$	Sec 3.4: Attack 4
full 16	Tweak all rounds	2^{55}	$2^{17} RK\text{-}CP$	-	[2]
full 16	Tweak all rounds	2^{55}	$2^{33} RK\text{-}KP$	2^{32}	[2]

$\star$ Worse than exhaustive search.

this as an important turning point towards efforts to break down its resistance against the slide attacks. A direct countermeasure is to add more irregularity to DES key schedule by having different shift values (not just 1 and 2) such that the Theorems 1 to 4 would no longer hold.

The realigning slide is another way to extend the slide attacks. Possible extensions could involve more complex techniques of sliding in order to penetrate key schedules with more subtle self-similarities, plus studying how the slide attacks can overcome the commonly-used slide attack countermeasure: round-dependence [5,6].

Acknowledgement

We thank the anonymous referees and Jongsung Kim for helpful comments and suggestions. This work is an extension of basic ideas sketched in [4]. We thank Alex Biryukov for stimulating discussions, David Wagner for providing a copy of his manuscript [22]; and to both Alex and David for their slide attacks [5,6] without which this paper would not have come into existence. We thank God for His many blessings (Dt 8:13). Part of this work was done while on vacation with wife and in-laws in coastal town of Bintulu, Sarawak.

References

1. E. Biham and A. Shamir: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology, Vol. 4, No. 1 (1991) 3–72
2. E. Biham: New Types of Cryptanalytic Attacks Using Related Keys. Journal of Cryptology, Vol. 7 (1994) 229–246
3. A. Biryukov: Methods of Cryptanalysis. Ph.D. Dissertation, Technion, Israel (1999)
4. A. Biryukov and R.C.-W. Phan: Extended Slide Attacks – Double and Realigning Slides. Unpublished manuscript, 2002
5. A. Biryukov and D. Wagner: Slide Attacks. Proceedings of Fast Software Encryption '99, LNCS 1636, Springer-Verlag (1999) 245–259
6. A. Biryukov and D. Wagner: Advanced Slide Attacks. Proceedings of Eurocrypt '00, LNCS 1807, Springer-Verlag (2000) 589–606
7. S. Furuya: Slide Attacks with a Known-Plaintext Cryptanalysis. Proceedings of ICISC '01, LNCS 2288, Springer-Verlag (2002) 214–225
8. H. Handschuh and D. Naccache: SHACAL. Submission to the NESSIE project (2000) Available from `http://www.cryptonessie.org`
9. H. Handschuh and D. Naccache: SHACAL: A Family of Block Ciphers. Submission to the NESSIE project (2002) Available from `http://www.cryptonessie.org`
10. S. Kavut, M.D. Yücel: Slide Attack on Spectr-H64. Proceedings of Indocrypt '02, LNCS 2551, Springer-Verlag (2002) 34–47
11. J. Kelsey, B. Schneier and D. Wagner: Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. Proceedings of Crypto '96, LNCS 1109, Springer-Verlag (1996) 237–251
12. J. Kelsey, B. Schneier and D. Wagner: Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2 and TEA. Proceedings of ICICS '97, LNCS 1334, Springer-Verlag (1997) 233–246

13. J. Kilian and P. Rogaway: How to Protect DES Against Exhaustive Key Search. Proceedings of Crypto '96, LNCS 1109, Springer-Verlag (1994) 252–267
14. J. Kilian and P. Rogaway: How to Protect DES Against Exhaustive Key Search (an Analysis of DESX). Journal of Cryptology, Vol. 14, No. 1 (2001) 17–35
15. L.R. Knudsen: New Potentially 'Weak' Keys for DES and LOKI (Extended abstract). Proceedings of Eurocrypt '94, LNCS 950, Springer-Verlag (1994) 419–424
16. L.R. Knudsen: Block Ciphers – Analysis, Design and Applications. PhD Thesis, Aarhus University, Denmark (1994)
17. M. Matsui: Linear Cryptanalysis Method for DES Cipher. Proceedings of Eurocrypt '93, LNCS 765, Springer-Verlag (1993) 386–397
18. NBS: Data Encryption Standard, Federal Information Processing Standard (FIPS), Publication 46, U.S. Dept. of Commerce, Washington D.C., January 1977
19. P. Onions: On the Strength of Simply-Iterated Feistel Ciphers with Whitening Keys. Proceedings of CT-RSA '01, LNCS 2020, Springer-Verlag (2001) 63–69
20. R.C.-W. Phan, S. Furuya: Sliding Properties of the DES Key Schedule and Potential Extensions to the Slide Attacks. Proceedings of ICISC '02, LNCS 2587, Springer-Verlag (2003) 138–148
21. M.-J.O. Saarinen: Cryptanalysis of Block Ciphers Based on SHA-1 and MD5. Proceedings of Fast Software Encryption '03, LNCS 2887, Springer-Verlag (2003) 36–44
22. D. Wagner: A Slide Attack on SHA-1. Unpublished manuscript, June 4, 2001

New Multiset Attacks on Rijndael with Large Blocks

Jorge Nakahara Jr.[1], Daniel Santana de Freitas[2], and Raphael C.-W. Phan[3]

[1] UniSantos, Brazil
jorge_nakahara@yahoo.com.br
[2] LabSEC, INE, Federal University of Santa Catarina, Brazil
santana@inf.ufsc.br
[3] iSECURES Lab, Swinburne University of Technology (Sarawak Campus), Malaysia
rphan@swinburne.edu.my

Abstract. This paper presents the first security evaluation of the **Rijndael cipher with block sizes larger than 128 bits**. We describe new higher-order multiset distinguishers for such large-block instances of Rijndael. Both Rijndael and the AES were designed to resist differential and linear cryptanalysis, which is indicated by the number of active S-boxes (minimum of 25 for 4-round AES) for the best differential and linear distinguishers, for which the probability and correlation values are estimated as 2^{-150} and 2^{-75}. All of these Rijndael variants have been formally defined by their designers as extensions of the AES. We describe new 5-round distinguishers for Rijndael with 160 up to 256-bit blocks, all holding with certainty, and with many more than 25 active S-boxes.

Keywords: Rijndael, higher-order multiset attacks, cryptanalysis.

1 Introduction

Rijndael is an SPN-type cipher designed by J. Daemen and V. Rijmen for the AES Development Process [17]. Both the block and key sizes can range from 128 up to 256 bits in steps of 32 bits [6, p.42]. The number of rounds is variable, depending on the block and key lengths. There are 25 instances of Rijndael formally defined by their designers [6,13] for all possible combinations of key and block sizes (Table 1). The 128-bit block version of Rijndael is officially known as the AES [17]. The other variants will be denoted Rijndael-160, Rijndael-192, Rijndael-224 and Rijndael-256, with the suffix indicating the block size in bits. The text and key blocks are usually represented by a $4 \times t$ state matrix of bytes, $4 \leq t \leq 8$. For instance, the state matrix for a $4t$-byte ($32t$-bit) text block, $A = (a_0, a_1, a_2, a_3, a_4, \ldots, a_{4t-1})$, is

$$\text{State} = \begin{pmatrix} a_0 & a_4 & \cdots & a_{4t-4} \\ a_1 & a_5 & \cdots & a_{4t-3} \\ a_2 & a_6 & \cdots & a_{4t-2} \\ a_3 & a_7 & \cdots & a_{4t-1} \end{pmatrix}, \tag{1}$$

namely, with the bytes filled columnwise. The AES has been extensively analyzed since 1997 but the same cannot be said of the other variants, most probably because they were not standardized as the AES. Rijndael-256, with a 256-bit key, had its software performance evaluated in the NESSIE Project [16], but there was no security analysis. However, analysis of these sisters of the AES may shed further light into the design of the AES and its structure as well as resistance against cryptanalysis.

This paper describes higher-order multiset distinguishers that consist, and therefore trace, the status of 128-bit words, instead of bytes as in [5]. Since our attacks require sets of 2^{128} chosen plaintexts at a time, the attacks do not apply to the AES, whose codebook size is 2^{128}. Nonetheless, this paper presents the first security evaluation of Rijndael with block sizes larger than 128 bits.

Table 1. Parameters of the Rijndael block cipher [6]

		Cipher				
		AES	**Rijndael-160**	**Rijndael-192**	**Rijndael-224**	**Rijndael-256**
Nr (# rounds)		Nb (# 32-bit words)				
		4	5	6	7	8
Nk (# 32-bit words)	4	10	11	12	13	14
	5	11	11	12	13	14
	6	12	12	12	13	14
	7	13	13	13	13	14
	8	14	14	14	14	14
ShiftRows Offsets	C_1	1	1	1	1	1
	C_2	2	2	2	2	3
	C_3	3	3	3	4	4

There are four layers in a full round transformation in Rijndael: AddRoundKey ($\mathbf{AK}_i$), SubBytes ($\mathbf{SB}_i$), ShiftRows ($\mathbf{SR}_i$) and MixColumns ($\mathbf{MC}_i$), all of which will be referred to as quarters of a round, or 0.25-round, so that distinguishers and attacks can be described more precisely. The subscripts i indicate the round number. One full round of Rijndael consists of $\mathrm{AK}_i \circ \mathrm{MC}_i \circ \mathrm{SR}_i \circ \mathrm{SB}_i(X) = \mathrm{AK}_i(\mathrm{MC}_i(\mathrm{SR}_i(\mathrm{SB}_i(X))))$, namely instantiation is in right-to-left order. There is an input transformation, AK_0 prior to the first round, and the last round does not include MC_i. For further details about Rijndael components refer to [6].

The paper is organized as follows: Sect. 2 gives basic definitions for the multiset attack. Sect. 3.1 describes attacks on Rijndael-160. Sect. 3.2 describes attacks on Rijndael-192. Sect. 3.3 describes attacks on Rijndael-224. Sect. 3.4 describes attacks on Rijndael-256. Sect. 4 concludes the paper.

2 Preliminaries

The multiset technique [2] has similarities with the Square attack [5], the saturation attack [14] and with integral cryptanalysis [10,12]. All of these techniques operate in a chosen-plaintext (CP) setting, and the first published one was a dedicated attack on the Square block cipher [5]. Nonetheless, this technique has already been applied to several ciphers, with or without wordwise operations [7,11,12,14]. A fundamental concept in a multiset attack is the Λ-set [5], which is a multiset [2] (a set with multiplicities) containing b full n-bit text block elements, where n is the block size and b is typically a power of 2. These n-bit text blocks are analysed by tracing fixed (but not necessarily contiguous) w bits, $w < n$, of all the text block elements. For example,

$$\{(0|1|2|3),(1|2|2|1),(3|1|2|2),(2|2|2|1),(7|5|2|0),(4|5|2|7),(5|5|2|4),(6|5|2|4)\}, \quad (2)$$

is a multiset with $2^w = 8$ elements, each of which is a 12-bit text block ($n = 12$). We further consider each of these 2^w elements as a concatenation of four w-bit words ($w = 3$), and thus, keep track of particular patterns in these w-bit words for each of the 2^w elements. When we focus only on certain fixed w bits of each element of the multiset, then we have w-bit sets having the same number of elements as the original multiset, but whose elements are formed by taking w bits at fixed positions in each text block of the original multiset. Often times, the w bits are composed of contiguous bits that respect word boundaries, hence the more common term word. The possible patterns in words are the following:

– if the w-bit elements of a word in a multiset assume each of the values 0 to $2^w - 1$, then the word is called a permutation or an active word [5], and is denoted 'P', e.g., the word formed by the first 3 bits in the multiset (2), which contains $\{0,1,3,2,7,4,5,6\}$;
– if all w-bit elements of a word in a multiset assume an arbitrary constant value, it is called passive or constant [5], and is denoted 'C', e.g., the third set of 3 bits in (2), which is $\{2,2,2,2,2,2,2,2\}$;
– if all w-bit elements of a word in a multiset occur an even number of times (each element has even multiplicity), it is called even, and is denoted 'E', e.g., the second set of 3 bits in (2), which is $\{1,2,1,2,5,5,5,5\}$;
– w-bit words which are either 'P' or 'E' are called dual, and are denoted 'D';
– if the sum of all w-bit values in a given word of a multiset, under some operator $\boxdot$, results in a predictable amount, then this word is called balanced, and is denoted 'B';
– otherwise, if the $\boxdot$-sum results in an unpredictable value, the word is called unbalanced, and is denoted '?', e.g., the fourth set of 3 bits in (2), which is $\{3,1,2,1,0,7,4,4\}$, with $\boxdot = \oplus$ i.e. exclusive-or.

The rationale behind the multiset technique is to use balanced sets of bits to attack permutation mappings (cipher rounds and its bijective components). Thus, multiset attacks exploit the bijective nature of internal cipher components. In particular, ciphers that operate on neatly partitioned words are the main targets. A typical multiset attack starts with multisets in which all words are balanced (usually only 'P' and 'C') and the propagation of balanced words in the multisets across multiple rounds of a cipher is traced up to the point in which the multiset is composed only of unbalanced bits. Rijndael uses basically three operations: bitwise exclusive-or, an 8×8 S-box, and multiplication in GF(2^8) =

$\mathrm{GF}(2)[x]/(x^8+x^4+x^3+x+1)$ called xtime. All of these operations are bijective. Since the exclusive-or ($\oplus$) operator is used to combine round subkeys with intermediate cipher data, it is natural to use $\oplus = \boxdot$ as the operator for computing the sum, because this value becomes independent of the round subkeys. Therefore, this sum is key-invariant, and thus also called an invariant. The expected exclusive-or sum for balanced words is zero. All multiset attacks reported in the literature have used either the $\oplus$ or the modular additive sum [10,18] as the operator in distinguishers. From initially checking all bits [5] for the 'B' pattern, other researchers have proposed to consider just one bit [14] or some (truncated) set of bits [12]. Other forms of sums have also been considered, depending on the particular cipher, such as $\boxdot = (\oplus, \boxplus, \boxplus, \oplus, \oplus, \boxplus, \boxplus, \oplus, \oplus, \boxplus, \boxplus, \oplus, \oplus, \boxplus, \boxplus, \oplus)$ in [10], where $\boxplus$ is addition modulo 256. One may also check for internal collisions, as considered by [9]. Essentially, we remark that the sums we check for depend on how many rounds a predictable pattern ('C', 'P', 'E', 'B', 'D') survives before becoming unpredictable ('?'). The multiset distinguishers in this paper all hold with certainty (probability one). The search for probabilistic multiset distinguishers is left as an open problem.

An interesting aspect about a multiset distinguisher is the number of active S-boxes[1], a concept inherited from differential and linear cryptanalysis [4]. An active S-box in a multiset distinguisher has both its input and output either 'D' or 'B', which are associated with non-zero (non-trivial) differences and masks in the case of differential and linear cryptanalysis. The 'C' and '?' words are associated with zero (or trivial) differences and linear masks. Similar to the differential and linear cases, the S-box is the main component that hinders the propagation of balanced words, in the sense that 'B' (but not 'D') words at the input to an S-box do not necessarily result in 'B' words at the output. Moreover, the S-box is often the only non-linear cipher component. It is interesting to observe that the number of active S-boxes in multiset distinguishers is relatively high compared to differential and linear distinguishers (Tables 7 and 8). Moreover, the former hold with certainty while the latter hold with much lower probability. All of these facts motivated our analysis of multiset distinguishers. Multisets containing 2^w text blocks, where w is 'typical' word size of the cipher (e.g. $w = 8$ for Rijndael), are designated 1st-order multisets. Further, multisets consisting of 2^{mw} text blocks each of which has m w-bit words, with some patterns, are designated mth-order multisets [12].

The following lemma and theorem will support the description of our multiset distinguishers:

Lemma 1. *For any w-bit 'P' word X, any subset of (not necessarily consecutive) y bits ($0 < y \leq w$) of X contains each y-bit value repeated 2^{w-y} times.*

Theorem 1. *For any w-bit dual word X, any subset of (not necessarily consecutive) y bits ($0 < y \leq w$) of X is even.*

[1] Do not confuse it with an active word.

Proof. Let X be 'E' i.e. every value in X repeats an even number of times. Pick all repetitions of one such value, say $x = x_0|x_1|\ldots|x_{w-1}$. Thus, x occurs in X $2y$ times, for some $y \geq 0$. Select any subset of bits x_{i_1}, x_{i_2}, ..., x_{i_y} of x. Since x repeats an even number of times, $x_{i_1}|x_{i_2}|\ldots|x_{i_y}$ also repeats an even number of times, from the even number of occurrences of x. Now, let X be 'P'. According to Lemma 1, any y bits of X contain each y-bit value repeated 2^{w-y} times, which is an even number. That concludes the proof.

3 Multiset Attacks on Rijndael with Blocks Larger Than 128 Bits

The propagation of multisets in higher-order distinguishers for Rijndael can be interpreted as the evolution of multisets across internal cipher components, either taken bytewise, or w-bitwise for $w > 8$. The existence of 1st-order distinguishers for Rijndael with blocks larger than 128 bits is not surprising at all, and can be explained exactly as for the AES. Thus, we describe the multisets and related attacks, but leave them without proof (further discussion will be in the full paper version [15]). We refer to [6] for further details. We have also considered 4th-order distinguishers that use 32-bit words instead of bytes as the next (intuitive) extension of the 1st-order distinguishers. The motivation for the latter is the MC matrix that has a 32-bit input. They are new for Rijndael variants, but were already described for the AES by Ferguson *et al.* in [8] so we present them without proof, and leave details to the full version [15]. Therefore, we focus primarily on **new** 16th-order distinguishers, that are motivated by the combination of SR_i and MC_i (two diffusion components): while MC_i operates on four bytes at a time, $\text{MC}_i \circ \text{SR}_i$ collects four bytes from a set of sixteen bytes (a 128-bit word). The 16th-order multiset distinguishers reach 5.25 rounds, allowing us to cross the 4-round barrier, as dictated by the cipher countermeasures against conventional differential and linear attacks. Moreover, another unique approach in the 16th-order distinguishers is that the multisets overlap, namely, they share columns of the state matrix.

Concerning the terminology, key-recovery attacks on r rounds of a cipher using an (r-j)-round distinguisher are denoted jR attacks, because subkeys of j rounds not covered by the distinguisher are recovered. In the key-recovery attacks described on Rijndael, we assume, as is common, that the MC_i layer is absent from the last round, even though we do not attack the full cipher. Even if MC_i were present, it could be undone efficiently in all chosen ciphertexts, because it is a fixed and key-independent transformation. Moreover, MC_i is linear with respect to AK_i, so it can be swapped with AK_i.

3.1 Rijndael-160

Multiset Distinguishers. For Rijndael-160, 3.25-round 1st-order distinguishers exist similar to the ones for the AES. One such distinguisher is detailed in the 2nd column of Table 3, where the multisets stand for the output multisets, namely after each 0.25-round. There are $\binom{20}{1} = 20$ such distinguishers, each one starting with a single 'P' plaintext byte (in a fixed byte position) and nineteen 'C' bytes.

A 4th-order distinguisher can start with a multiset in which the four plaintext bytes a_0, a_5, a_{10}, a_{15} (state matrix (1) with $t = 5$ columns) form a 32-bit 'P' word, while the remaining bytes are 'C'. There are $\binom{5}{4} = 5$ possible such distinguishers. The multisets in each quarter round are listed in the 3rd column of Table 3. Each multiset consists of five 32-bit words (each column of the state).

The new 16th-order multiset distinguisher is described in the 4th column of Table 3. There are $\binom{5}{4} = 5$ different such distinguishers, and in each of which there is a 128-bit 'P' word consisting of four columns as input to MC_1. Without loss of generality, we use one with a multiset in which the plaintext bytes a_0, a_2, a_3, a_4, a_5, a_7, a_8, a_9, a_{10}, a_{12}, a_{13}, a_{14}, a_{15}, a_{17}, a_{18} and a_{19} form a 128-bit 'P' word, while the remaining bytes are 'C'. Thus, this distinguisher consists of a 5-tuple of overlapping words, each tracing the behavior of four consecutive columns of the state (in a circular fashion): $a_0|a_1|\ldots|a_{15}$, $a_4|a_5|\ldots|a_{19}$, $a_8|a_9|\ldots|a_3$, $a_{12}|a_{13}|\ldots|a_7$ and $a_{16}|a_{17}|\ldots|a_{11}$. This approach is unique to the 16th-order distinguisher, and allows us to trace 5-tuples of 128-bit words at once, showing patterns that would not be evident otherwise. We believe it makes sense to trace the status of overlapping words because 128 is not a factor of 160. Without the overlapping-word approach we would have to report the status of a 128-bit and a 32-bit word (160 bits in total). The plaintext multiset, as well as the one after AK_0, has the form (**E E E E E**), where each **E** stands for the status of a 128-bit word consisting of four consecutive columns of the state. That is because each word contains at least one 'C' byte. The same multiset remains after SB_1. After SR_1, the multiset becomes (**P E E E E**), as expected, due to the choice of the plaintext multiset. After MC_1, AK_1 and SB_2 the multiset remains (**P E E E E**) since parallel MC matrices, xor with a fixed subkey, and parallel application of a fixed S-box are bijective mappings. SR_2 permutes the bytes of the 128-bit 'P' word such that each four consecutive columns of the state contain at least three 'C' bytes (while the remaining bytes are even). Thus, the output after SR_2 becomes (**E E E E E**). This same multiset remains after MC_2, after AK_2, and after SB_3, since they are bijective mappings, either bytewise or 128-bitwise. SR_3 splits the even bytes to different columns of the state. It is proved algebraically in Appendix B that the multiset after SR_3 has the form (**E E P E E**). Next, since the parallel application of four MC matrices, the xor with a fixed subkey, and the parallel application of a fixed S-box are all bijective mappings, the multiset (**E E P E E**) remains after MC_3, AK_3, and SB_4. SR_4 splits the even bytes to different columns of the state, resulting in the multiset (**B B B B B**). The reasoning is that the bytes that constitute four consecutive columns of the state are not necessarily 'E', because they come from different 128-bit 'D' words. Nonetheless, using Theorem 1, the 32-bit inputs to each MC matrix in MC_4 come from the same 128-bit 'D' word (due to the SR offsets), and are thus even. Since the inputs to each MC in MC_4 are even, the 32-bit outputs (each column) of each MC in MC_4 are also even. This explains why the multiset (**B B B B B**) can propagate across AK_4 and SB_5 (128-bit words are 'B', but 32-bit words are 'E'). SR_5 splits the even bytes output from SB_5 to different MC matrices in MC_5. The combination $MC_5\circ$ SR_5 causes the patterns at the byte level to deteriorate in the sense that the individual bytes are not 'E' anymore, but simply 'B'. The resulting multiset after MC_5 as well as after AK_5, then has the form (**B B B B B**) both wordwise and bytewise. Thus, after SB_6, the resulting multiset becomes (**? ? ? ? ?**) because 'B' bytes cannot generally propagate across SB_6. Thus, the distinguisher reaches 5.25 rounds, from AK_0 until AK_5.

Multiset Attacks. The 3.25-round 1st-order distinguisher described in the 2nd column of Table 3 allows a 1R attack on 4-round Rijndael-160. We guess subkey K_4 bytewise, **decrypt** $SB_4\circ$ SR_4 and check the zero xor sum after AK_3 bytewise. The attack complexity is $2 \cdot 2^8 = 2^9$ CP (to avoid false alarms because the xor sum is tested on 8-bit values), and $2^8 \cdot 2^8 = 2^{16}$ S-box computations per K_4 byte, or $20 \cdot 2^{16}/(2 \cdot 4) \approx 2^{17}$ 4-round computations (because $SB_4\circ$ SR_4 accounts for 1/2 of a round) for the full 20-byte K_4. A 1st-order distinguisher can also be used in a 2R attack on 5-round Rijndael-160. In this case, one recovers four bytes from K_5 and one byte from K_4 at once, by partially decrypting $SB_4\circ$ $SR_4\circ$ $MC_4\circ$ $AK_4\circ$ $SB_5\circ$ $SR_5\circ$ AK_5 until after AK_3, and checking for the zero xor sum bytewise. The attack complexity is $6 \cdot 2^8$ CP (to avoid false alarms) and $2^8 \cdot 2^{40} + 2^8 \cdot 2^{32} + 2^8 \cdot 2^{24} + 2^8 \cdot 2^{16} + 2^8 \cdot 2 + 2^8 \cdot 1 \approx 2^{48}$ S-box computations to recover five subkey bytes, or $5 \cdot 2^{48}/(7 \cdot 5) \approx 2^{45}$ 5-round computations to recover the full K_5 and five bytes from K_4. Other attack extensions to 6-round Rijndael-160 could simply guess the full AK_0, decrypt the first round and apply the previous attack on 5 rounds, but the effort increases to $2^{160} \cdot 2^{45} = 2^{205}$ 6-round computations.

The 4.25-round 4th-order distinguisher described in the 3rd column of Table 3 allows a 1R attack on 5-round Rijndael-160. We guess subkey K_5 bytewise, partially decrypt $SB_5\circ$ SR_5 and check the zero xor sum after AK_4 for the correct subkey value. The attack complexity is $2 \cdot 2^{32} = 2^{33}$ CP (to avoid false alarms), and $2^{32} \cdot 2^8 = 2^{40}$ S-box computations per K_5 byte, or $20 \cdot 2^{40}/(2 \cdot 5) = 2^{41}$ 5-round computations (because $SB_5\circ$ SR_5 accounts for 1/2 of a round) for the full 20-byte K_5. A 4th-order distinguisher can also be used in a 2R attack on 6-round Rijndael-160. In this case, one recovers four bytes from K_6 and one byte from K_5 at once by partially decrypting $SB_5\circ$ $SR_5\circ$ $MC_5\circ$ $AK_5\circ$ $SB_6\circ$ $SR_6\circ$ AK_6 until right after AK_4, and checking for the zero xor sum. The attack complexity is $6 \cdot 2^{32}$ CP and $2^{32} \cdot 2^{40} + 2^{32} \cdot 2^{32} + 2^{32} \cdot 2^{24} + 2^{32} \cdot 2^{16} + 2^{32} \cdot 2^8 + 2^{32} \approx 2^{72}$ S-box computations, to recover five subkey bytes, or $5 \cdot 2^{72} \cdot 1/7 \cdot 1/6 \approx 2^{69}$ 6-round computations, to recover the full K_6 and five bytes from K_5. Further, a 3R attack on 7-round Rijndael-160 could simply guess AK_0, for instance, and apply the previous attack on the resulting 6 rounds. The attack complexity becomes $2^{160} \cdot 2^{69} = 2^{229}$ 7-round computations, and $6 \cdot 2^{32}$ CP.

The 5.25-round 16th-order distinguisher in the 4th column of Table 3 allows a 1R attack on 6-round Rijndael-160. One guesses subkey K_6 bytewise, decrypts $SB_6\circ$ SR_6, and checks the zero xor sum after AK_5 for the correct subkey value.

The attack complexity is $2 \cdot 2^{128} = 2^{129}$ CP (to avoid false alarms) and $2^{128} \cdot 2^8 = 2^{136}$ S-box computations per K_6 byte, or $20 \cdot 2^{128}/(2 \cdot 6) \approx 2^{129}$ 6-round computations for the full K_6. A 16th-order distinguisher can also be used in a 2R attack on 7-round Rijndael-160. In this case, one recovers four bytes from K_7 and one byte from K_6 at once by partially decrypting $SB_6\circ$ $SR_6\circ$ $MC_6\circ$ $AK_6\circ$ $SB_7\circ$ $SR_7\circ$ AK_7, until after AK_5, and checking for the zero xor sum. The attack complexity is $6 \cdot 2^{128}$ CP (to avoid false alarms) and $2^{128} \cdot 2^{40} + 2^{128} \cdot 2^{32} + 2^{128} \cdot 2^{24} + 2^{128} \cdot 2^{16} + 2^{128} \cdot 2^8 + 2^{128} \approx 2^{168}$ S-box computations to recover five subkey bytes. These computations, though, can be grouped more efficiently, reducing the complexity significantly, using Ferguson *et al.* partial-sum technique [8]. This technique reduces the complexity to $4 \cdot 2^{128} \cdot 2^{16} = 2^{146}$ S-box computations to recover five subkey bytes, or $5 \cdot 2^{146}/(4 \cdot 7) \approx 2^{144}$ 7-round computations, to recover the full K_7 and four byte from K_6. This attack works similarly to the one of Ferguson *et al.* on the AES. We refer to [8] for further details.

3.2 Rijndael-192

Multiset Distinguishers. The 2nd column of Table 4 details the propagation of multisets in a 1st-order distinguisher for which the plaintext byte a_0 is 'P', while the remaining bytes are 'C' (state matrix (1) with $t = 7$). The 3rd column of Table 4 details the propagation of multisets in a 4th-order multiset distinguisher in which plaintext bytes a_0, a_5, a_{10}, and a_{15} form a 32-bit 'P' word, while the other bytes are 'C'. Each symbol in this multiset stands for a 32-bit word (one column of the state). The propagation of multisets can be explained similarly to that of the AES [8,12], but taking into account the 192-bit block size. The 3rd column of Table 4 details the propagation of multisets in a 16th-order distinguisher in which the plaintext bytes a_0, a_3, a_4, a_5, a_8, a_9, a_{10}, a_{12}, a_{13}, a_{14}, a_{15}, a_{17}, a_{18}, a_{19}, a_{22} and a_{23} form a 128-bit 'P' word, while the remaining bytes are 'C'. This distinguisher consists of 6-tuples of overlapping 128-bit words that traces the behaviour of four consecutive columns of the state (in a circular fashion): $a_0|a_1|\ldots|a_{15}$, $a_4|a_5|\ldots|a_{19}$, $a_8|a_9|\ldots|a_{23}$, $a_{12}|a_{13}|\ldots|a_3$, $a_{16}|a_{17}|\ldots|a_7$, and $a_{20}|a_{21}|\ldots|a_{11}$. The propagation of multisets can be explained similarly to the one on Rijndael-160.

Multiset Attacks. The 3.25-round 1st-order distinguisher described in the 2nd column of Table 4 allows a 1R attack on 4-round Rijndael-192. We guess subkey K_4 bytewise, decrypt $SB_4\circ$ SR_4 and check the zero xor sum after AK_3. The attack complexity is $2 \cdot 2^8 = 2^9$ CP (to avoid false alarms), and $2^8 \cdot 2^8 = 2^{16}$ S-box computations per K_4 byte, or $24 \cdot 2^{16}/(2 \cdot 4) \approx 2^{17}$ 4-round computations (because $SB_4\circ$ SR_4 accounts for 1/2 of a round). Further, a 2R attack on 5-round Rijndael-192 can recover four bytes from K_5 and one byte from K_4 at once. Again, the attack partially decrypts $SB_4\circ$ $SR_4\circ$ $MC_4\circ$ $AK_4\circ$ $SB_5\circ$ $SR_5\circ$ AK_5 until after AK_3, and checks for the zero xor sum. The attack complexity is $6 \cdot 2^8$ CP (to avoid false alarms) and $2^8 \cdot 2^{40} + 2^8 \cdot 2^{32} + 2^8 \cdot 2^{24} + 2^8 \cdot 2^{16} + 2^8 \cdot 2 + 2^8 \cdot 1 \approx 2^{48}$ S-box computations to recover five subkey bytes, or $6 \cdot 2^{48}/(7 \cdot 5) \approx 2^{46}$ 5-round computations to recover K_5 and six bytes from K_4. Finally, a 3R attack on 6-round Rijndael-192 can simply guess one full subkey and apply the previous attack on 5 rounds. The attack complexity is $6 \cdot 2^8$ CP, and $2^{192} \cdot 2^{46} = 2^{238}$ 6-round computations.

The 4th-order distinguisher in the 3rd column of Table 4 allows a 1R attack on 5-round Rijndael-192. We guess subkey bytes K_5^i for $i \in \{0, 1, 2, 3, 4, 5, 6, 8, 9, 12, 15, 18, 19, 21, 22, 23\}$ and check for the zero xor sum after AK_4. The remaining subkeys cannot be recovered using this distinguisher, because of the rightmost (**E E**) at the end of the distinguisher. It requires another 4th-order distinguisher e.g. one for which plaintext bytes a_8, a_{13}, a_{18}, and a_{23} form a 32-bit 'P' word, and the remaining bytes are 'C'. The attack complexity is $2 \cdot 2^{32} = 2^{33}$ CP and $2^8 \cdot 2^{32} = 2^{40}$ S-box computations per subkey byte, or $24 \cdot 2^{40}/(2 \cdot 5) = 2^{41}$ 5-round computations. Further, a 4th-order distinguisher can be used in 2R attacks on 6-round Rijndael-192. In this case, one recovers four bytes from K_6 and one byte from K_5 at once. This attack partially decrypts $SR_5\circ$ $MC_5\circ$ $AK_5\circ$ $SB_6\circ$ $SR_6\circ$ AK_6 until the rightmost 64-bit output of SB_5 in the distinguisher which has the form **(E E)**. The attack complexity is $6 \cdot 2^{32}$ CP and $2^{32} \cdot 2^{40} + 2^{32} \cdot 2^{32} + 2^{32} \cdot 2^{24} + 2^{32} \cdot 2^{16} + 2^{32} \cdot 2^8 + 2^{32} \approx 2^{72}$ S-box computations. Using Ferguson *et al.* partial-sum technique [8] can reduce this complexity to $4 \cdot 2^{32} \cdot 2^{16} = 2^{50}$ S-box computations to recover five subkey bytes. Or, $5 \cdot 2^{50}/(6 \cdot 6) \approx 2^{47}$ 6-round computations (partial $SR_5\circ$ $MC_5\circ$ $AK_5\circ$ $SB_6\circ$ $SR_6\circ$ AK_6 computation account for 1/6 of a round). Finally, a 3R attack on 7-round Rijndael-192 can simply guess one full subkey, and apply the previous attack on 6 rounds, at the cost of $6 \cdot 2^{32}$ CP, and $2^{192} \cdot 2^{47} = 2^{239}$ 7-round computations.

The 5.25-round 16th-order distinguisher in the 4th column of Table 4 allows a 1R attack on 6-round Rijndael-192. Guess subkey K_6 bytewise, partially decrypt $SB_6\circ$ SR_6, and check for the zero xor sum after AK_5, for the correct subkey value. The attack complexity is $2 \cdot 2^{128} = 2^{129}$ CP, and $2^{128} \cdot 2^8 = 2^{136}$ S-box computations per K_6 byte, or $24 \cdot 2^{136}/(2 \cdot 6) = 2^{137}$ 6-round computations. Further, a 2R attack on 7-round Rijndael-192 works similar to the same attack on Rijndael-160. Using the partial-sum optimization technique in [8] results in $6 \cdot 2^{128}$ CP and $4 \cdot 2^{128} \cdot 2^{16} = 2^{146}$ S-box computations to recover five subkey bytes. Or, $6 \cdot 2^{146} \cdot 1/4 \cdot 1/7 \approx 2^{144}$ 7-round computations.

3.3 Rijndael-224

Multiset Distinguishers. The 2nd column of Table 5 details the propagation of multisets in a 1st-order multiset distinguisher for which the plaintext byte a_0 is 'P' while the remaining bytes are 'C'. The 3rd column of Table 5 details the propagation of multisets in a 4th-order multiset distinguisher in which the plaintext bytes a_0, a_5, a_{10}, and a_{19} form a 32-bit 'P' word, while the remaining bytes are 'C'. The multisets consist of seven 32-bit words (seven columns of the state). The propagation of multisets can be explained similarly to that of the AES [8,12], but taking into account the larger block size, and the different SR offsets (Table 1). The 4th column of Table 5 details the propagation of multisets in a 16th-order distinguisher in which the plaintext bytes a_0, a_4, a_5, a_7, a_8, a_9, a_{10}, a_{13}, a_{14}, a_{16}, a_{18}, a_{19}, a_{21}, a_{23}, a_{26}, and a_{27} form a 128-bit 'P' word. The propagation of multisets can be explained similarly to Rijndael-160 and Rijndael-192, but taking into account the larger block size, and the different SR offsets (Table 1).

Multiset Attacks. The 3.75-round 1st-order distinguisher in the 2nd column of Table 5 allows a 1R attack on 4-round Rijndael-224. Subkey bytes K_4^0, K_4^{15}, K_4^{22}, and K_4^{25} cannot be determined using this distinguisher, because the output of SR_4 is 'P' for these subkey bytes. To recover them, another 1st-order distinguisher has to be used, e.g. one in which plaintext byte a_1 is 'P' while the remaining bytes are 'C'. It does not affect the overall complexity. We guess a subkey byte K_4^i, $i \in \mathbb{Z}_{28} - \{0, 15, 22, 25\}$, decrypt $SB_4\circ$ SR_4, and check the zero xor sum for the correct guess. The attack complexity is $2 \cdot 2^8 = 2^9$ CP and $2^8{\cdot}2^8 = 2^{16}$ S-box computations per subkey byte recovered, or $28{\cdot}2^{16}/(2{\cdot}4) \approx 2^{18}$ 4-round computations for the full 28-byte K_4. Additionally, a 2R attack on 5-round Rijndael-224 can recover four bytes from K_5 and one byte from K_4 at once. The attack partially decrypts $SR_4\circ$ $MC_4\circ$ $AK_4\circ$ $SB_5\circ$ $SR_5\circ$ AK_5 until the leftmost 32-bit output from SB_4, where the correct subkey value can be checked. The attack complexity is $6 \cdot 2^8$ CP and about $2^8 \cdot 2^{40} = 2^{48}$ S-box computations, or $7 \cdot 2^{48}/(6 \cdot 5) \approx 2^{46}$ 5-round computations to recover the full K_5 and seven bytes from K_4. A 3R attack could guess one full subkey and apply the previous attack on 5 rounds, at the cost of $6 \cdot 2^8$ CP and $2^{46+192} = 2^{238}$ 6-round computations.

The 4.25-round 4th-order distinguisher in the 3rd column of Table 5 allows a 1R attack on 5-round Rijndael-224. Guess K_5 bytewise, decrypt $SB_5\circ$ SR_5 and check the zero xor sum after AK_4 for the correct subkey value. The attack complexity is $2 \cdot 2^{32} = 2^{33}$ CP (to avoid false alarms), and $2^{32} \cdot 2^8 = 2^{40}$ S-box computations per K_5 byte, or $28 \cdot 2^{40}/(2 \cdot 5) \approx 2^{42}$ 5-round computations (because $SB_5\circ$ SR_5 accounts for 1/2 of a round) for the full 28-byte K_5. This distinguisher can also be used in a 2R attack on 6-round Rijndael-192. In this cases, one recovers four bytes from K_6 and one byte from K_5 at once. This attack partially decrypts $SB_5\circ$ $SR_5\circ$ $MC_5\circ$ $AK_5\circ$ $SB_6\circ$ $SR_6\circ$ AK_6 until after AK_4, and checks for the zero xor sum. The attack complexity is $6 \cdot 2^{32}$ CP and $2^{32} \cdot 2^{40} + 2^{32} \cdot 2^{32} + 2^{32} \cdot 2^{24} + 2^{32} \cdot 2^{16} + 2^{32} \cdot 2^8 + 2^{32} \approx 2^{72}$ S-box computations, to recover five subkey bytes, or $7 \cdot 2^{72}/(7 \cdot 6) \approx 2^{70}$ 6-round computations, to recover the full K_6 and seven bytes from K_5.

The 5.25-round 16th-order distinguisher in the 4th column of Table 5 allows a 1R attack on 6-round Rijndael-224. Guess K_6 bytewise, decrypt $SB_6\circ$ SR_6 and check the zero xor sum after AK_5. The attack complexity is $2{\cdot}2^{128} = 2^{129}$ CP, and $2^{128}{\cdot}2^8 = 2^{136}$ S-box computations per K_6 byte, or $28{\cdot}2^{136}/(2{\cdot}6) \approx 2^{137}$ 6-round computations. Further, a 2R attack on 7-round Rijndael-224 works similarly to the same attack on Rijndael-160. Using the partial-sum technique in [8] results in $6 \cdot 2^{128}$ CP and 2^{146} S-box computations to recover five subkey bytes, or $7 \cdot 2^{146}/(4 \cdot 7) = 2^{144}$ 7-round computations to recover the full K_7, and seven bytes from K_6.

3.4 Rijndael-256

Multiset Distinguishers. The 2nd column of Table 6 details the propagation of multisets in a 1st-order multiset distinguisher in which the plaintext byte a_0 is 'P', while the other bytes are 'C'. The 3rd column of Table 6 details the propagation of multisets in a 4th-order distinguisher in which the plaintext bytes a_0, a_5, a_{14}, a_{19} form a 32-bit 'P' word, while the remaining bytes are 'C'. Each symbol in these multisets consists of a 32-bit word (a column of the state), and their propagation can be explained similarly to those of the AES [8,12], but taking into account the larger block size, and the different SR offsets (Table 1). The 4th column of Table 6 details the propagation of multisets in a 16th-order distinguisher in which the plaintext bytes a_0, a_3, a_4, a_5, a_9, a_{12}, a_{14}, a_{16}, a_{17}, a_{18}, a_{19}, a_{21}, a_{23}, a_{26}, a_{30}, and a_{31} form a 32-bit 'P' word, while the remaining bytes are 'C'.

Multiset Attacks. The 3.75-round 1st-order distinguisher in the 2nd column of Table 6 allows a 1R attack on 4-round Rijndael-256. The attack is similar to the one on Rijndael-224. The attack complexities are $2 \cdot 2^8 = 2^9$ CP and $2^8 \cdot 2^8 = 2^{16}$ S-box computations per subkey byte, or $32 \cdot 2^{16}/(2 \cdot 4) = 2^{18}$ 4-round computations. Additionally, a 2R attack on 5-round Rijndael-256 can recover four bytes from K_5 and one byte from K_4 at once, similar to the same attack on Rijndael-224. The attack complexities are $6 \cdot 2^8$ CP and about $2^8 \cdot 2^{40} = 2^{48}$ S-box computations, or $8 \cdot 2^{48}/(6 \cdot 5) \approx 2^{46}$ 5-round computations to recover the full K_5 and seven bytes from K_4.

The 4.25-round 4th-order distinguisher in the 3rd column of Table 6 allows a 1R attack on 5-round Rijndael-256, similar to the same attack on Rijndael-224. The attack complexity is $2 \cdot 2^{32} = 2^{33}$ CP, and $2^{32} \cdot 2^8 = 2^{40}$ S-box computations, or $32 \cdot 2^{40}/(2 \cdot 5) \approx 2^{41}$ 5-round computations ($SB_5 \circ SR_5$ accounts for 1/2 of a round). A 4th-order distinguisher can also be used in a 2R attack on 6-round Rijndael-256, similar to the same attack on Rijndael-224. The attack complexity is $6 \cdot 2^{32}$ CP and $2^{32} \cdot 2^{40} + 2^{32} \cdot 2^{32} + 2^{32} \cdot 2^{24} + 2^{32} \cdot 2^{16} + 2^{32} \cdot 2^8 + 2^{32} \approx 2^{72}$ S-box computations, or $8 \cdot 2^{72}/(7 \cdot 6) \approx 2^{70}$ 6-round computations.

The 5.25-round 16th-order distinguisher in the 4th column of Table 6 allows a 1R attack on 6-round Rijndael-256, similar to the same attack on Rijndael-224. The attack complexity is $2 \cdot 2^{128} = 2^{129}$ CP, and $2^{128} \cdot 2^8 = 2^{136}$ S-box computations per K_6 byte, or $32 \cdot 2^{136}/(2 \cdot 6) \approx 2^{137}$ 6-round computations for the full 32-byte K_6. Further, a 2R attack on 7-round Rijndael-256 works similar to the same attack on Rijndael-160. Using the partial-sum technique in [8] results in $6 \cdot 2^{128}$ CP and 2^{146} S-box computations to recover five subkey bytes, or $8 \cdot 2^{146}/(4 \cdot 8) = 2^{144}$ 7-round computations to recover the full K_7, and seven bytes from K_6.

4 Conclusions and Open Problems

The multiset attacks described in this paper considered higher-order multisets, i.e., using n-bit words, for $n \in \{8, 32, 128\}$, instead of bytes as in [5] (the latter is known as a 1st-order attack). The nth-order multisets were chosen because of the MC_i and the $\text{MC}_i \circ \text{SR}_i$ transformations. Our main finding is that higher-order multiset distinguishers become 1-round longer as we increase the attack order in powers of four (Table 8). Moreover, the distinguishers also become slightly longer for larger block sizes, because of the slower diffusion due to MC and the different SR offsets (Table 1). For comparison purposes, Table 7 lists the corresponding minimum number of active S-boxes in differential and linear distinguishers for Rijndael for the same number of rounds as the multiset distinguishers in Table 8. The slight difference in the size of 4th-order distinguishers for Rijndael-192 (4.50 rounds) compared to the other variants (4.25 rounds) is due to the SR offsets. Recall from Table 1 that Rijndael-$32t$, $4 \leq t \leq 6$, have the same SR offsets, but Rijndael-224 and Rijndael-256 have different offsets. Our attacks do not apply to the AES, since we require one or more sets of 2^{128} CP.

Table 2. Multiset attack complexities for Rijndael with blocks larger than 128 bits

Cipher	# Rounds	Key Sizes	Data (CP)	Memory	Time	Attack
Rijndael-160	4	all	2^9	2^8	2^{17}	1st-order Multiset
	5	all	2^{33}	2^{32}	2^{41}	4th-order Multiset
(160-bit	5	all	$6 \cdot 2^8$	2^8	2^{45}	1st-order Multiset
block)	6	all	$6 \cdot 2^{32}$	2^{32}	2^{69}	4th-order Multiset
	6	> 128	2^{129}	2^{128}	2^{129}	16th-order Multiset
	6	224; 256	$6 \cdot 2^8$	2^8	2^{205}	1st-order Multiset
	7	> 128	$6 \cdot 2^{128}$	2^{128}	2^{144}	16th-order Multiset
	7	256	$6 \cdot 2^{32}$	2^{32}	2^{229}	4th-order Multiset
Rijndael-192	4	all	2^9	2^8	2^{17}	1st-order Multiset
	5	all	2^{33}	2^{32}	2^{41}	4th-order Multiset
(192-bit	5	all	$6 \cdot 2^8$	2^8	2^{46}	1st-order Multiset
block)	6	all	$6 \cdot 2^{32}$	2^{32}	2^{47}	4th-order Multiset
	6	> 128	2^{129}	2^{128}	2^{137}	16th-order Multiset
	7	> 128	$6 \cdot 2^{128}$	2^{128}	2^{144}	16th-order Multiset
	7	256	$6 \cdot 2^8$	2^8	2^{238}	1st-order Multiset
	7	256	$6 \cdot 2^{32}$	2^{32}	2^{239}	4th-order Multiset
Rijndael-224	4	all	2^9	2^8	2^{18}	1st-order Multiset
	5	all	2^{33}	2^{32}	2^{42}	4th-order Multiset
(224-bit	5	all	$6 \cdot 2^8$	2^8	2^{46}	1st-order Multiset
block)	6	all	$6 \cdot 2^{32}$	2^{32}	2^{70}	4th-order Multiset
	6	> 128	2^{129}	2^{128}	2^{137}	16th-order Multiset
	6	256	$6 \cdot 2^8$	2^8	2^{238}	1st-order Multiset
	7	> 128	$6 \cdot 2^{128}$	2^{128}	2^{144}	16th-order Multiset
Rijndael-256	4	all	2^9	2^8	2^{18}	1st-order Multiset
	5	all	2^{33}	2^{32}	2^{41}	4th-order Multiset
(256-bit	5	all	$6 \cdot 2^8$	2^8	2^{46}	1st-order Multiset
block)	6	all	$6 \cdot 2^{32}$	2^{32}	2^{70}	4th-order Multiset
	6	> 128	2^{129}	2^{128}	2^{137}	16th-order Multiset
	7	> 128	$6 \cdot 2^{128}$	2^{128}	2^{144}	16th-order Multiset

Notable properties of higher-order multiset distinguishers are: the relatively large number of active S-boxes compared to their differential and linear counterparts, and the size of the distinguishers themselves, which are longer than 1st-order ones (Table 8). This may be an evidence of as yet unknown multiset cryptanalytic properties worth further study. Table 2 compares the attack complexities on reduced-round Rijndael for block sizes larger than 128 bits. The distinguishers in Tables 3, 4, 5 and 6 are independent of the key and subkey values (no weak-key assumption), and of the key schedule algorithms.

Acknowledgements

The first author thanks Siang W. Song and Bart Preneel for the continued encouragement along all of this research. The third author thanks Lars Knudsen for general inspiration and for introducing the first ever multiset-style attack [5]. We thank the anonymous referees (past and present) for their comments and interest in this work.

References

1. E. Biham, N. Keller, "Cryptanalysis of Reduced Variants of Rijndael," 3rd AES Conference, New York, USA, 2000, http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html
2. A. Biryukov, A. Shamir, "Structural Cryptanalysis of SASAS," Advances in Cryptology, Eurocrypt'01, B. Pfitzmann, Ed., Springer-Verlag, LNCS 2045, 2001, 394–405.
3. J.H. Cheon, M. Kim, K. Kim, J.-Y. Lee, S.W. Kang, "Improved Impossible Differential Cryptanalysis of Rijndael and Crypton," Proceedings of ICISC 2001, K. Kim, Ed., Springer Verlag, LNCS 2288, 2001, 39–49.
4. D. Coppersmith, "The Data Encryption Algorithm and its Strength Against Attacks," IBM Journal on Research and Development (38):3, 1994, 243–250.
5. J. Daemen, L.R. Knudsen, V. Rijmen, "The Block Cipher SQUARE,"4th Fast Software Encryption Workshop, E. Biham, Ed., Springer-Verlag, LNCS 1267, 1997, 149–165.
6. J. Daemen, V. Rijmen, "The Design of Rijndael – AES – The Advanced Encryption Standard," Springer-Verlag, 2002.
7. H. Demirci, "Square-like Attacks on Reduced Rounds of IDEA," 9th Selected Areas in Cryptography Workshop, SAC'02, K. Nyberg, H. Heys, Eds., Springer-Verlag, LNCS 2595, Aug, 2002, 147–159.
8. N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, D. Whiting, "Improved Cryptanalysis of Rijndael," 7th Fast Software Encryption Workshop, B. Schneier, Ed., Springer-Verlag, LNCS 1978, 2000, 213–230.
9. H. Gilbert, M. Minier, "A Collision Attack on Seven Rounds of Rijndael," 3rd AES Conference, New York, USA, 2000, http://csrc.nist.gov/encryption/aes/

10. Y. Hu, Y. Zhang, G. Xiao, "Integral Cryptanalysis of SAFER+," Electronic Letters, vol. 35, number 17, Aug. 1999, 1458–1459.
11. I. Kim, Y. Yeom, H. Kim, "Square Attacks on the Reduced-Round MISTY1," SCIS, Symposium on Cryptography and Information Security, Jan, 2002, 921–924.
12. L.R. Knudsen, D. Wagner, "Integral Cryptanalysis," 9th Fast Software Encryption Workshop, J. Daemen and V. Rijmen, Eds., Springer-Verlag, LNCS 2365, 2002, 112–127.
13. H.W. Lenstra, "Rijndael for Algebraists," Apr. 8, 2002, *http://math.berkeley.* `edu/ hwl/papers/rijndael0.pdf`
14. S. Lucks, "The Saturation Attack – a Bait for Twofish," 8th Fast Software Encryption Workshop, M. Matsui, Ed., Springer-Verlag, LNCS 2355, 2001, 1–15.
15. J. Nakahara Jr., D.S. de Freitas, R.C.-W. Phan, "New Multiset Attacks on Rijndael with Large Blocks," Full version of this paper, 2005.
16. NESSIE, "New European Schemes for Signatures, Integrity and Encryption," Jan. 2000, http://cryptonessie.org.
17. NIST, "Advanced Encryption Standard AES," FIPS PUB 197 Federal Information Processing Standard Publication 197, U.S. Department of Commerce, Nov, 2001.
18. G. Piret, J.-J. Quisquater, "Integral Cryptanalysis on Reduced-round Safer++: A way to extend the attack?", NESSIE Public Report, NES/DOC/UCL/WP5/002/1, 2003.

Appendix A

This appendix details the distinguishers for Rijndael ciphers with blocks larger than 128 bits, in Tables 3 to 6.

Table 3. Multiset distinguishers for Rijndael-160

Layer	1st-order multisets	4th-order multisets	16th-order multisets
AK_0	(P C C C C C C C C C C C C C C C C C C C)	(E E E E C)	(E E E E E)
SB_1	(P C C C C C C C C C C C C C C C C C C C)	(E E E E C)	(E E E E E)
SR_1	(P C C C C C C C C C C C C C C C C C C C)	(P C C C C)	(P E E E E)
MC_1	(P P P P C C C C C C C C C C C C C C C C)	(P C C C C)	(P E E E E)
AK_1	(P P P P C C C C C C C C C C C C C C C C)	(P C C C C)	(P E E E E)
SB_2	(P P P P C C C C C C C C C C C C C C C C)	(P C C C C)	(P E E E E)
SR_2	(P C C C C C C C C C C P C C P C C P C C)	(E C E E E)	(E E E E E)
MC_2	(P P P P C C C C P P P P P P P P P P P P)	(E C E E E)	(E E E E E)
AK_2	(P P P P C C C C P P P P P P P P P P P P)	(E C E E E)	(E E E E E)
SB_3	(P P P P C C C C P P P P P P P P P P P P)	(E C E E E)	(E E E E E)
SR_3	(P C P P C P P P P P P P P P P C P P C P)	(E E P E E)	(E E P E E)
MC_3	(B B B B B B B B B B B B B B B B B B B B)	(E E P E E)	(E E P E E)
AK_3	(B B B B B B B B B B B B B B B B B B B B)	(E E P E E)	(E E P E E)
SB_4	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(E E P E E)	(E E P E E)
SR_4	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(B B B B B)	(B B B B B)
MC_4	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(B B B B B)	(B B B B B)
AK_4	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(B B B B B)	(B B B B B)
SB_5	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(? ? ? ? ?)	(B B B B B)
SR_5	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(? ? ? ? ?)	(B B B B B)
MC_5	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(? ? ? ? ?)	(B B B B B)
AK_5	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(? ? ? ? ?)	(B B B B B)
SB_6	(? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?)	(? ? ? ? ?)	(? ? ? ? ?)

Table 4. Multiset distinguishers for Rijndael-192

Layer	1st-order multisets	4th-order multisets	16th-order multisets
AK_0	(P C C C C C C C C C C C C C C C C C C C C C C C)	(E E E E C C)	(E E E E E E)
SB_1	(P C C C C C C C C C C C C C C C C C C C C C C C)	(E E E E C C)	(E E E E E E)
SR_1	(P C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C)	(P E E E E E)
MC_1	(P P P P C C C C C C C C C C C C C C C C C C C C)	(P C C C C C)	(P E E E E E)
AK_1	(P P P P C C C C C C C C C C C C C C C C C C C C)	(P C C C C C)	(P E E E E E)
SB_2	(P P P P C C C C C C C C C C C C C C C C C C C C)	(P C C C C C)	(P E E E E E)
SR_2	(P C C C C C C C C C C C C C C P C C P C C P C C)	(E C C E E E)	(E E E E E E)
MC_2	(P P P P C C C C C C C C P P P P P P P P P P P P)	(E C C E E E)	(E E E E E E)
AK_2	(P P P P C C C C C C C C P P P P P P P P P P P P)	(E C C E E E)	(E E E E E E)
SB_3	(P P P P C C C C C C C C P P P P P P P P P P P P)	(E C C E E E)	(E E E E E E)
SR_3	(P C C P C C P P C P P P P P P P P P P C P P C C)	(E E E P E E)	(E E E P E E)
MC_3	(B B B B B B B B B B B B B B B B B B B B B B B B)	(E E E P E E)	(E E E P E E)
AK_3	(B B B B B B B B B B B B B B B B B B B B B B B B)	(E E E P E E)	(E E E P E E)
SB_4	(? ?)	(E E E P E E)	(E E E P E E)
SR_4	(? ?)	(B B B B E E)	(B B B B B B)
MC_4	(? ?)	(B B B B E E)	(B B B B B B)
AK_4	(? ?)	(B B B B E E)	(B B B B B B)
SB_5	(? ?)	(? ? ? ? E E)	(B B B B B B)
SR_5	(? ?)	(? ? ? ? ? ?)	(B B B B B B)
MC_5	(? ?)	(? ? ? ? ? ?)	(B B B B B B)
AK_5	(? ?)	(? ? ? ? ? ?)	(B B B B B B)
SB_6	(? ?)	(? ? ? ? ? ?)	(? ? ? ? ? ?)

Table 5. Multiset distinguishers for Rijndael-224

Layer	1st-order multisets	4th-order multisets	16th-order multisets
AK_0	(P C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(E E E C E C C)	(E E E E E E E)
SB_1	(P C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(E E E C E C C)	(E E E E E E E)
SR_1	(P C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C C)	(P E E E E E E)
MC_1	(P P P P C C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C C)	(P E E E E E E)
AK_1	(P P P P C C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C C)	(P E E E E E E)
SB_2	(P P P P C C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C C)	(P E E E E E E)
SR_2	(P C C C C C C C C C C C C C C P C C C C C C P C C P C C)	(E C C E C E E)	(E E E E E E E)
MC_2	(P P P P C C C C C C C C P P P P C C C C P P P P P P P P)	(E C C E C E E)	(E E E E E E E)
AK_2	(P P P P C C C C C C C C P P P P C C C C P P P P P P P P)	(E C C E C E E)	(E E E E E E E)
SB_3	(P P P P C C C C C C C C P P P P C C C C P P P P P P P P)	(E C C E C E E)	(E E E E E E E)
SR_3	(P C C C C C P P C P C P P C P P C P P C P P P C P P C P)	(E E E E E E E)	(E E E E E E E)
MC_3	(P P P P B B B B B B B B B B B B B B B B B B B B B B B B)	(E E E E E E E)	(E E E E E E E)
AK_3	(P P P P B B B B B B B B B B B B B B B B B B B B B B B B)	(E E E E E E E)	(E E E E E E E)
SB_4	(P P P P ?)	(E E E E E E E)	(E E E E E E E)
SR_4	(P ? ? ? ? ? ? ? ? ? ? ? ? ? ? P ? ? ? ? ? ? P ? ? P ? ?)	(B B B B B B B)	(B B B B B B B)
MC_4	(? ?)	(B B B B B B B)	(B B B B B B B)
AK_4	(? ?)	(B B B B B B B)	(B B B B B B B)
SB_5	(? ?)	(? ? ? ? ? ? ?)	(B B B B B B B)
SR_5	(? ?)	(? ? ? ? ? ? ?)	(B B B B B B B)
MC_5	(? ?)	(? ? ? ? ? ? ?)	(B B B B B B B)
AK_5	(? ?)	(? ? ? ? ? ? ?)	(B B B B B B B)
SB_6	(? ?)	(? ? ? ? ? ? ?)	(? ? ? ? ? ? ?)

Table 6. Multiset distinguishers for Rijndael-256

Layer	1st-order multisets	4th-order multisets	16th-order multisets
AK_0	(P C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(E E C E E C C C)	(E E E E E E E E)
SB_1	(P C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(E E C E E C C C)	(E E E E E E E E)
SR_1	(P C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C C C)	(P E E E E E E E)
MC_1	(P P P P C C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C C C)	(P E E E E E E E)
AK_1	(P P P P C C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C C C)	(P E E E E E E E)
SB_2	(P P P P C C C C C C C C C C C C C C C C C C C C C C C C C C C C)	(P C C C C C C C)	(P E E E E E E E)
SR_2	(P C C C C C C C C C C C C C C C C C C P C C P C C C C C C P C C)	(E C C C E E C E)	(E E E E E E E E)
MC_2	(P P P P C C C C C C C C C C C C P P P P P P P P C C C C P P P P)	(E C C C E E C E)	(E E E E E E E E)
AK_2	(P P P P C C C C C C C C C C C C P P P P P P P P C C C C P P P P)	(E C C C E E C E)	(E E E E E E E E)
SB_3	(P P P P C C C C C C C C C C C C P P P P P P P P C C C C P P P P)	(E C C C E E C E)	(E E E E E E E E)
SR_3	(P C C P C C P P C C P C C P C P P P P P P C P C C P C C P P C C)	(E E E E P E E E)	(E E E E P E E E)
MC_3	(B B B B B B B B P P P P B B B B B B B B B B B B P P P P B B B B)	(E E E E P E E E)	(E E E E P E E E)
AK_3	(B B B B B B B B P P P P B B B B B B B B B B B B P P P P B B B B)	(E E E E P E E E)	(E E E E P E E E)
SB_4	(? ? ? ? ? ? ? ? P P P P ? ? ? ? ? ? ? ? ? ? ? ? P P P P ? ? ? ?)	(E E E E P E E E)	(E E E E P E E E)
SR_4	(? ? ? ? ? P ? ? P ? ? P ? ? P ? ? ? ? ? ? P ? ? P ? ? P ? ? P ?)	(B B B B B B B B)	(B B B B B B B B)
MC_4	(? ?)	(B B B B B B B B)	(B B B B B B B B)
AK_4	(? ?)	(B B B B B B B B)	(B B B B B B B B)
SB_5	(? ?)	(? ? ? ? ? ? ? ?)	(B B B B B B B B)
SR_5	(? ?)	(? ? ? ? ? ? ? ?)	(B B B B B B B B)
MC_5	(? ?)	(? ? ? ? ? ? ? ?)	(B B B B B B B B)
AK_5	(? ?)	(? ? ? ? ? ? ? ?)	(B B B B B B B B)
SB_6	(? ?)	(? ? ? ? ? ? ? ?)	(? ? ? ? ? ? ? ?)

Appendix B

This appendix details a proof of the propagation of the 16th-order multisets across the 1.25-round transformation $T = \mathrm{SR}_3 \circ \mathrm{SB}_3 \circ \mathrm{AK}_2 \circ \mathrm{MC}_2 \circ \mathrm{SR}_2$ in Rijndael-160, in Sect. 3.1. In this case, the input multiset to T has the form (**P E E E E**). More precisely, let the input to T be denoted $A = (a_0, a_1, \ldots, a_{18}, a_{19})$. Thus, $a_0|a_1|\ldots|a_{14}|a_{15}$ is a 128-bit 'P' word, while $a_{16}|a_{17}|a_{18}|a_{19}$ is a 32-bit 'C' word. The SR_i transformation can be represented as a multiplication of all bytes $(a_0, \ldots, a_{19})$ of the input state A with a 20×20 permutation matrix:

$$SR = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0
\end{pmatrix} . \tag{3}$$

The intermediate value, $U = (u_0, u_1, \ldots, u_{19}) = MC_2 \circ SR_2(A)$, consists of a further multiplication with:

$$MC = \begin{pmatrix}
2&3&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&2&3&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&2&3&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
3&1&1&2&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&2&3&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&2&3&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&2&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&3&1&1&2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&2&3&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&2&3&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&2&3&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&3&1&1&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&2&3&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&2&3&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&2&3&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&3&1&1&2&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&2&3&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&2&3&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&2&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&3&1&1&2
\end{pmatrix} , \tag{4}$$

resulting in

$$\begin{aligned}
U = (&2a_0 \oplus 3a_5 \oplus a_{10} \oplus a_{15}, a_0 \oplus 2a_5 \oplus 3a_{10} \oplus a_{15}, a_0 \oplus a_5 \oplus 2a_{10} \oplus 3a_{15}, 3a_0 \oplus a_5 \oplus a_{10} \oplus 2a_{15},\\
&2a_4 \oplus 3a_9 \oplus a_{14} \oplus a_{19}, a_4 \oplus 2a_9 \oplus 3a_{14} \oplus a_{19}, a_4 \oplus a_9 \oplus 2a_{14} \oplus 3a_{19}, 3a_4 \oplus a_9 \oplus a_{14} \oplus 2a_{19}\\
&2a_8 \oplus 3a_{13} \oplus a_{18} \oplus a_3, a_8 \oplus 2a_{13} \oplus 3a_{18} \oplus a_3, a_8 \oplus a_{13} \oplus 2a_{18} \oplus 3a_3, 3a_8 \oplus a_{13} \oplus a_{18} \oplus 2a_3,\\
&2a_{12} \oplus 3a_{17} \oplus a_2 \oplus a_7, a_{12} \oplus 2a_{17} \oplus 3a_2 \oplus a_7, a_{12} \oplus a_{17} \oplus 2a_2 \oplus 3a_7, 3a_{12} \oplus a_{17} \oplus a_2 \oplus 2a_7\\
&2a_{16} \oplus 3a_1 \oplus a_6 \oplus a_{11}, a_{16} \oplus 2a_1 \oplus 3a_6 \oplus a_{11}, a_{16} \oplus a_1 \oplus 2a_6 \oplus 3a_{11}, 3a_{16} \oplus a_1 \oplus a_6 \oplus 2a_{11}) .
\end{aligned} \tag{5}$$

Finally, $T(A) = SR_3 \circ SB_3 \circ AK_2(U)$ can be represented as

$$\begin{aligned}T(A) = \ & (S[2a_0 \oplus 3a_5 \oplus a_{10} \oplus a_{15} \oplus K_2^0], S[a_4 \oplus 2a_9 \oplus 3a_{14} \oplus a_{19} \oplus K_2^5],\\ & S[a_8 \oplus a_{13} \oplus 2a_{18} \oplus 3a_3 \oplus K_2^{10}], S[3a_{12} \oplus a_{17} \oplus a_2 \oplus 2a_7 \oplus K_2^{15}],\\ & S[2a_4 \oplus 3a_9 \oplus a_{14} \oplus a_{19} \oplus K_2^4], S[a_8 \oplus 2a_{13} \oplus 3a_{18} \oplus a_3 \oplus K_2^9],\\ & S[a_{12} \oplus a_{17} \oplus 2a_2 \oplus 3a_7 \oplus K_2^{14}], S[3a_{16} \oplus a_1 \oplus a_6 \oplus 2a_{11} \oplus K_2^{19}],\\ & S[2a_8 \oplus 3a_{13} \oplus a_{18} \oplus a_3 \oplus K_2^8], S[a_{12} \oplus 2a_{17} \oplus 3a_2 \oplus a_7 \oplus K_2^{13}],\end{aligned} \tag{6}$$

$$\begin{aligned}& S[a_{16} \oplus a_1 \oplus 2a_6 \oplus 3a_{11} \oplus K_2^{18}], S[3a_0 \oplus a_5 \oplus a_{10} \oplus 2a_{15} \oplus K_2^3],\\ & S[2a_{12} \oplus 3a_{17} \oplus a_2 \oplus a_7 \oplus K_2^{12}], S[a_{16} \oplus 2a_1 \oplus 3a_6 \oplus a_{11} \oplus K_2^{17}],\\ & S[a_0 \oplus a_5 \oplus 2a_{10} \oplus 3a_{15} \oplus K_2^2], S[3a_4 \oplus a_9 \oplus a_{14} \oplus 2a_{19} \oplus K_2^7],\\ & S[2a_{16} \oplus 3a_1 \oplus a_6 \oplus a_{11} \oplus K_2^{16}], S[a_0 \oplus 2a_5 \oplus 3a_{10} \oplus a_{15} \oplus K_2^1],\\ & S[a_4 \oplus a_9 \oplus 2a_{14} \oplus 3a_{19} \oplus K_2^6], S[3a_8 \oplus a_{13} \oplus a_{18} \oplus 2a_3 \oplus K_2^{11}]) .\end{aligned} \tag{7}$$

We prove by contradiction that for the input multiset A of the form (**P E E E E**), the output multiset of $T(A)$ has the form (**E E P E E**). The main contention might be the sole 128-bit 'P' word in the middle of the multiset. The 'E' words can be justified similarly. Suppose otherwise, that is, the middle word is not 'P'. Then, some value in $T(A)$, wordwise, might repeat if it is not actually a permutation of all 128-bit values. Without loss of generality, consider just two 128-bit values, $t, t' \in T(A)$, corresponding to the 'P' word, namely the 1st, 3rd, 4th and 5th columns of the state matrix, and such that $t = t'$ for distinct text inputs a and a' from A. Thus,

$$\begin{aligned}t = \ & (S[2a_8 \oplus 3a_{13} \oplus a_{18} \oplus a_3 \oplus K_2^8], S[a_{12} \oplus 2a_{17} \oplus 3a_2 \oplus a_7 \oplus K_2^{13}],\\ & S[a_{16} \oplus a_1 \oplus 2a_6 \oplus 3a_{11} \oplus K_2^{18}], S[3a_0 \oplus a_5 \oplus a_{10} \oplus 2a_{15} \oplus K_2^3],\\ & S[2a_{12} \oplus 3a_{17} \oplus a_2 \oplus a_7 \oplus K_2^{12}], S[a_{16} \oplus 2a_1 \oplus 3a_6 \oplus a_{11} \oplus K_2^{17}],\\ & S[a_0 \oplus a_5 \oplus 2a_{10} \oplus 3a_{15} \oplus K_2^2], S[3a_4 \oplus a_9 \oplus a_{14} \oplus 2a_{19} \oplus K_2^7],\\ & S[2a_{16} \oplus 3a_1 \oplus a_6 \oplus a_{11} \oplus K_2^{16}], S[a_0 \oplus 2a_5 \oplus 3a_{10} \oplus a_{15} \oplus K_2^1],\\ & S[a_4 \oplus a_9 \oplus 2a_{14} \oplus 3a_{19} \oplus K_2^6], S[3a_8 \oplus a_{13} \oplus a_{18} \oplus 2a_3 \oplus K_2^{11}],\\ & S[2a_0 \oplus 3a_5 \oplus a_{10} \oplus a_{15} \oplus K_2^0], S[a_4 \oplus 2a_9 \oplus 3a_{14} \oplus a_{19} \oplus K_2^5],\\ & S[a_8 \oplus a_{13} \oplus 2a_{18} \oplus 3a_3 \oplus K_2^{10}], S[3a_{12} \oplus a_{17} \oplus a_2 \oplus 2a_7 \oplus K_2^{15}]),\end{aligned}$$

and

$$\begin{aligned}t' = \ & (S[2a'_8 \oplus 3a'_{13} \oplus a'_{18} \oplus a'_3 \oplus K_2^8], S[a'_{12} \oplus 2a'_{17} \oplus 3a'_2 \oplus a'_7 \oplus K_2^{13}],\\ & S[a'_{16} \oplus a'_1 \oplus 2a'_6 \oplus 3a'_{11} \oplus K_2^{18}], S[3a'_0 \oplus a'_5 \oplus a'_{10} \oplus 2a'_{15} \oplus K_2^3],\\ & S[2a'_{12} \oplus 3a'_{17} \oplus a'_2 \oplus a'_7 \oplus K_2^{12}], S[a'_{16} \oplus 2a'_1 \oplus 3a'_6 \oplus a'_{11} \oplus K_2^{17}],\\ & S[a'_0 \oplus a'_5 \oplus 2a'_{10} \oplus 3a'_{15} \oplus K_2^2], S[3a'_4 \oplus a'_9 \oplus a'_{14} \oplus 2a'_{19} \oplus K_2^7],\\ & S[2a'_{16} \oplus 3a'_1 \oplus a'_6 \oplus a'_{11} \oplus K_2^{16}], S[a'_0 \oplus 2a'_5 \oplus 3a'_{10} \oplus a'_{15} \oplus K_2^1],\\ & S[a'_4 \oplus a'_9 \oplus 2a'_{14} \oplus 3a'_{19} \oplus K_2^6], S[3a'_8 \oplus a'_{13} \oplus a'_{18} \oplus 2a'_3 \oplus K_2^{11}],\\ & S[2a'_0 \oplus 3a'_5 \oplus a'_{10} \oplus a'_{15} \oplus K_2^0], S[a'_4 \oplus 2a'_9 \oplus 3a'_{14} \oplus a'_{19} \oplus K_2^5],\\ & S[a'_8 \oplus a'_{13} \oplus 2a'_{18} \oplus 3a'_3 \oplus K_2^{10}], S[3a'_{12} \oplus a'_{17} \oplus a'_2 \oplus 2a'_7 \oplus K_2^{15}]).\end{aligned}$$

In particular, equality holds bytewise, that is, $S[2a_0 \oplus 3a_5 \oplus a_{10} \oplus a_{15} \oplus K_2^0] = S[2a'_0 \oplus 3a'_5 \oplus a'_{10} \oplus a'_{15} \oplus K_2^0]$, and similarly for the other fifteen byte values. Since $a_0|a_1|\ldots|a_{14}|a_{15}$ in A is a 128-bit 'P' word, and $a_{16}|a_{17}|a_{18}|a_{19}$ is a 'C' word, and the subkeys are fixed, these equalities can be expressed as$S[2a_0 \oplus 3a_5 \oplus a_{10} \oplus a_{15} \oplus c_0] = S[2a'_0 \oplus 3a'_5 \oplus a'_{10} \oplus a'_{15} \oplus c'_0]$, and similarly for the other 15 bytes, where c_i and c'_i, $0 \leq i \leq 15$ are constants. But, $c_i = c'_i$ because they depend on the same subkeys. Therefore, $S[2a_0 \oplus 3a_5 \oplus a_{10} \oplus a_{15} \oplus c_0] = S[2a'_0 \oplus 3a'_5 \oplus a'_{10} \oplus a'_{15} \oplus c_0]$, or $2a_0 \oplus 3a_5 \oplus a_{10} \oplus a_{15} = 2a'_0 \oplus 3a'_5 \oplus a'_{10} \oplus a'_{15}$, and similarly for the other 15 bytes, namely, $a_4 \oplus 2a_9 \oplus 3a_{14} = a'_4 \oplus 2a'_9 \oplus 3a'_{14}$, $a_8 \oplus a_{13} \oplus 3a_3 = a'_8 \oplus a'_{13} \oplus 3a'_3$, $3a_{12} \oplus a_2 \oplus 2a_7 = 3a'_{12} \oplus a'_2 \oplus 2a'_7$, $2a_8 \oplus 3a_{13} \oplus a_3 = 2a'_8 \oplus 3a'_{13} \oplus a'_3$, $a_{12} \oplus 3a_2 \oplus a_7 = a'_{12} \oplus 3a'_2 \oplus a'_7$, $a_1 \oplus 2a_6 \oplus 3a_{11} = a'_1 \oplus 2a'_6 \oplus 3a'_{11}$, $3a_0 \oplus a_5 \oplus a_{10} \oplus 2a_{15} = 3a'_0 \oplus a'_5 \oplus a'_{10} \oplus 2a_{15}$, $2a_{12} \oplus a_2 \oplus a_7 = 2a'_{12} \oplus a'_2 \oplus a'_7$, $2a_1 \oplus 3a_6 \oplus a_{11} = 2a'_1 \oplus 3a'_6 \oplus a'_{11}$, $a_0 \oplus a_5 \oplus 2a_{10} \oplus 3a_{15} = a'_0 \oplus a'_5 \oplus 2a'_{10} \oplus 3a'_{15}$, $3a_4 \oplus a_9 \oplus a_{14} = 3a'_4 \oplus a'_9 \oplus a'_{14}$, $3a_1 \oplus a_6 \oplus a_{11} = 3a'_1 \oplus a'_6 \oplus a'_{11}$, $a_0 \oplus 2a_5 \oplus 3a_{10} \oplus a_{15} = a'_0 \oplus 2a'_5 \oplus 3a'_{10} \oplus a'_{15}$, $a_4 \oplus a_9 \oplus 2a_{14} = a'_4 \oplus a'_9 \oplus 2a'_{14}$, $3a_8 \oplus a_{13} \oplus 2a_3 = 3a'_8 \oplus a'_{13} \oplus 2a'_3$. Solving $a_1 \oplus 2a_6 \oplus 3a_{11} = a'_1 \oplus 2a'_6 \oplus 3a'_{11}$, $2a_1 \oplus 3a_6 \oplus a_{11} = 2a'_1 \oplus 3a'_6 \oplus a'_{11}$ and $3a_1 \oplus a_6 \oplus a_{11} = 3a'_1 \oplus a'_6 \oplus a'_{11}$ via Gaussian elimination, results in $a_1 = a'_1$, $a_6 = a'_6$, and $a_{11} = a'_{11}$. Similarly, from $a_4 \oplus 2a_9 \oplus 3a_{14} = a'_4 \oplus 2a'_9 \oplus 3a'_{14}$, $a_4 \oplus a_9 \oplus 2a_{14} = a'_4 \oplus a'_9 \oplus 2a'_{14}$ and $3a_4 \oplus a_9 \oplus a_{14} = 3a'_4 \oplus a'_9 \oplus a'_{14}$ results in $a_4 = a'_4$, $a_9 = a'_9$, and $a_{14} = a'_{14}$. Similarly, from $a_8 \oplus a_{13} \oplus 3a_3 = a'_8 \oplus a'_{13} \oplus 3a'_3$, $2a_8 \oplus 3a_{13} \oplus a_3 = 2a'_8 \oplus 3a'_{13} \oplus a'_3$, and $3a_8 \oplus a_{13} \oplus 2a_3 = 3a'_8 \oplus a'_{13} \oplus 2a'_3$ results in $a_8 = a'_8$, $a_{13} = a'_{13}$, and $a_3 = a'_3$. Finally, from $2a_0 \oplus 3a_5 \oplus a_{10} \oplus a_{15} = 2a'_0 \oplus 3a'_5 \oplus a'_{10} \oplus a'_{15}$, $3a_0 \oplus a_5 \oplus a_{10} \oplus 2a_{15} = 3a'_0 \oplus a'_5 \oplus a'_{10} \oplus 2a_{15}$, $a_0 \oplus a_5 \oplus 2a_{10} \oplus 3a_{15} = a'_0 \oplus a'_5 \oplus 2a'_{10} \oplus 3a'_{15}$, and $a_0 \oplus 2a_5 \oplus 3a_{10} \oplus a_{15} = a'_0 \oplus 2a'_5 \oplus 3a'_{10} \oplus a'_{15}$, results in $a_0 = a'_0$, $a_5 = a'_5$, $a_{10} = a'_{10}$, and $a_{15} = a'_{15}$. In total $a_i = a'_i$, for $0 \leq i \leq 15$, but at least one of these equalities should not hold because $a_0|a_1|\ldots|a_{15}$ is a 'P' word. It is a contradiction. The justification for the 'E' words in A is that the equalities with the four variables a_0, a_5, a_{10}, and a_{15} involve only three equalities, which form an indeterminate system that can be solved uniquely by Gaussian elimination. Thus, the output of 1.25-round transformation T is the multiset (**E E P E E**).

Appendix C

This appendix shows Table 7 and 8, comparing the size of distinguishers and the number of active S-boxes in differential/linear attacks and the corresponding figures for multiset attacks.

Table 7. Number of active S-boxes in differential and linear distinguishers for Rijndael ciphers

Cipher	# Rounds	# Act. S-boxes	# Rounds	# Act. S-boxes	# Rounds	# Act. S-boxes
AES	3.25	9	4.25	25	—	—
Rijndael-160	3.25	9	4.25	25	5.25	45
Rijndael-192	3.25	9	4.50	49	5.25	49
Rijndael-224	3.75	25	4.25	53	5.25	53
Rijndael-256	3.75	25	4.25	57	5.25	67

Table 8. Number of active S-boxes in multiset distinguishers for Rijndael ciphers

Cipher	1st-order (# Rounds)	# Act. S-boxes	4th-order (# Rounds)	# Act. S-boxes	16th-order (# Rounds)	# Act. S-boxes
AES	3.25	21	4.25	40	—	—
Rijndael-160	3.25	21	4.25	44	5.25	92
Rijndael-192	3.25	21	4.50	56	5.25	104
Rijndael-224	3.75	25	4.25	52	5.25	106
Rijndael-256	3.75	29	4.25	56	5.25	128

Paillier's Cryptosystem Modulo p^2q and Its Applications to Trapdoor Commitment Schemes

Katja Schmidt-Samoa[1] and Tsuyoshi Takagi[2]

[1] Technische Universität Darmstadt, Fachbereich Informatik,
Hochschulstr. 10, D-64289 Darmstadt, Germany
samoa@informatik.tu-darmstadt.de

[2] Future University – Hakodate, School of Systems Information Science,
116-2 Kamedanakano-cho Hakodate, Hokkaido, 041-8655, Japan
takagi@fun.ac.jp

Abstract. In 1998/99, T. Okamoto and S. Uchiyama on the one hand and P. Paillier on the other hand introduced homomorphic encryption schemes semantically secure against passive adversaries (IND-CPA). Both schemes follow in the footsteps of Goldwasser-Micali, Benaloh-Fischer and Naccache-Stern cryptosystems, and yield their improvements above the latter by changing the group structure. Paillier's scheme works in the group $\mathbb{Z}_{n^2}^{\times}$ where n is an RSA modulus, whilst Okamoto-Uchiyama is located in the group $\mathbb{Z}_n^{\times}$ for n of p^2q type. The new schemes attracted much attention because of their rich mathematical structure. It is notable that Okamoto-Uchiyama is one-way under the p^2q factoring assumption, whilst there is no reduction known from the one-wayness of Paillier's scheme to a standard computational assumption.

In this paper we point out that the combination of both techniques yields a new scheme that inherits all the nice properties of Paillier's scheme and that is one-way under the p^2q factoring assumption. The one-wayness is based on a new trapdoor one-way function which might be of independent interest. In addition, we show how to construct trapdoor commitment schemes with practical applications based on our new scheme and on the trapdoor function. Among other things, we propose a trapdoor commitment scheme that perfectly meets the requirements to construct Shamir-Tauman on-line/off-line signatures.

Keywords: homomorphic encryption, trapdoor commitments, trapdoor hash families, on-line/off-line signatures, chameleon signatures

1 Introduction

In their seminal paper from 1984 Goldwasser and Micali introduced the notion of semantic security and presented the first cryptosystem meeting this requirements [GM84]. Their proposed cryptosystem is additively homomorphic and probabilistic but suffers from a very limited bandwidth (the encryption is performed bit-wise). Over the intervening years this scheme has been improved several times, where the most notable ameliorations came from Benaloh-Fischer

[CF85] and Naccache-Stern [NS98]. However, the actual breakthrough in the field of semantically secure additive homomorphic encryption has been achieved by Okamoto-Uchiyama and Paillier with a different approach. Namely, their idea was to change the group structure from $\mathbb{Z}_n^\times$ with a RSA modulus n to $\mathbb{Z}_n^\times$ with $n = p^2q$ (Okamoto-Uchiyama [OU98]), resp. $\mathbb{Z}_{n^2}^\times$ (Paillier [Pai99]). Both works gained recognition not only for presenting practical solutions to homomorphic encryption, but also for pointing out the rich mathematical structure of the groups $\mathbb{Z}_n^\times$ with $n = p^2q$ resp. $\mathbb{Z}_{n^2}^\times, n = pq$. Whilst the assumptions on which semantic security relies seems to be comparable for both schemes (*p-subgroup assumption* versus *decisional composite residuosity assumption*), this is not the case for one-wayness: Okamoto-Uchiyama's cryptosystem can be proven one-way if factoring integers p^2q is hard, but for Paillier's scheme no reduction to a standard intractability assumption has been observed yet[1].

Our Contributions. Our first contribution is the development of a factorization-based variant of Paillier's homomorphic encryption scheme. Our concept is to study Paillier's original encryption function in a different group, *i.e.* instead of $\mathbb{Z}_{n^2}^\times$ with an RSA modulus n we consider the group $\mathbb{Z}_{n^2}^\times$ with the Okamoto-Uchiyama modulus $n = p^2q$. Based on the analysis of a new trapdoor one-way function which we introduce in Sect. 2.2, we are able to show that the proposed cryptosystem is one-way under the p^2q factorization assumption. Moreover, the new scheme inherits all the nice properties of Paillier's original one, such as semantic security, additively homomorphic property and efficiency. Unfortunately, the new scheme inherits the most serious drawback of Okamoto-Uchiyama's cryptosystem, too, namely it is vulnerable to a simple chosen ciphertext attack (in general this seems to be the flip-side of the coin regarding factorization-based one-wayness, see *e.g.* textbook Rabin). Of course, there are standard techniques to overcome this problem, for instance the clever use of hash functions, but all in all we feel that this part of the paper is predominantly of theoretical value.

In the rest of the paper we develop practical applications of our novel scheme and the underlying one-way function. More precisely, we introduce two new trapdoor commitments intended as building blocks for Shamir-Tauman on-line/off-line signatures [ST01] and chameleon signatures [KR00]. In the case of on-line/off-line signatures our proposed trapdoor commitment scheme to the best of our knowledge is the first one to yield a highly efficient and perfectly powerful construction at the same time. In addition, we propose the first factorization-based trapdoor commitment that can be used to construct on-line/off-line chameleon signatures, therefore improving the RSA(n, n)-based construction from [CGHGN01]. For a more detailed motivation and comparison see Sect. 4. As the first part of the paper is indeed of theoretical interest, but achieves no significant improvement in homomorphic encryption, we regard our new trapdoor commitment schemes as our main contribution.

[1] In [Pai99], Paillier based the one-wayness of his scheme on the *composite residuosity assumption*, but this assumption is merely a paraphrase of the designated one-wayness property.

2 Preliminaries

2.1 Notations

Let n be a positive integer. We write $\mathbb{Z}_n$ for the ring of residue classes modulo n, and $\mathbb{Z}_n^\times$ for its multiplicative group.

We denote the image of a function f with $\mathrm{im}(f)$.

As usual, a probability $\Pr(k)$ is called *negligible* if $\Pr(k)$ decreases faster than the inverse of any polynomial in k, *i.e.* $\forall c \exists k_c (k > k_c \Rightarrow \Pr(k) < k_c^{-c})$. In contrast, a probability $\Pr(k)$ is called *overwhelming*, if $1 - \Pr(k)$ is negligible.

We abbreviate *probabilistic polynomial time algorithm* by PPA.

Finally, $|n|_2$ denotes the bit-length of the integer n, and we write $[n]^k$ for the integer corresponding to the k most significant bits of n.

2.2 Mathematical Foundations

Throughout this section, let p, q be primes with $p \nmid q-1$ and $q \nmid p-1$ and define $n = p^2 q$. In the following, some facts about the group structure of $\mathbb{Z}_n^\times$ resp. $\mathbb{Z}_{n^2}^\times$ are developed.

First, we define an equivalence relation on $\mathbb{Z}_n^\times$:

Definition 1 (The equivalence relation $\sim$). *We define $\sim$ on $\mathbb{Z}_n^\times$ as follows:*

$$x \sim y \iff x^n = y^n \bmod n.$$

It is easy to see that $\sim$ indeed defines an equivalence relation on $\mathbb{Z}_n^\times$. We can proof the following theorem:

Theorem 1. *For $x, y \in \mathbb{Z}_n^\times$ we have*

$$x \sim y \iff x = y \bmod pq.$$

Proof. "$\Leftarrow$": $(x + kpq)^n = \sum_{i=0}^n \binom{n}{i} x^{n-i} (kpq)^i = x^n + n x^{n-1} kpq = x^n \bmod n$.
"$\Rightarrow$": $x^n = y^n \bmod n \Rightarrow x^n = y^n \bmod pq \Rightarrow x = y \bmod pq$, because n and $\varphi(pq)$ are coprime.

□

Now we can easily describe the equivalence class of $x \in \mathbb{Z}_n^\times$:

Corollary 1. *Let us denote with $[x]$ the equivalence class to which $x \in \mathbb{Z}_n^\times$ belongs. Then we have:*

$$[x] = \{x + ipq \bmod n | 0 \le i < p\}.$$

Thus there is exactly one element in $[x]$ which is smaller than pq, namely $x \bmod pq$.

In the subsequent sections, we exploit the fact that the problem of determining this element for arbitrary $x \in \mathbb{Z}_n^\times$ is as hard as factoring n. In Appendix B, we discuss the hardness of factoring integers of the special form $n = p^2 q$.

Next, we define the quotient group $\mathbb{Z}_n^\times/_\sim$

Definition 2 (N-R(n))**.** *Let* $\text{N-R}(n) = \{x \in \mathbb{Z}_n^\times | x = y^n \bmod n \text{ for a } y \in \mathbb{Z}_n^\times\}$ *denote the set of the* n*th residues modulo* n*.*

N-R(n) is a subgroup of $\mathbb{Z}_n^\times$ of order $(p-1)(q-1)$ (due to the fact that there are exactly $\varphi(pq) = (p-1)(q-1)$ pairwise different nth residues modulo n, namely the elements $\{x^n \bmod n | x \in \mathbb{Z}_{pq}^\times\}$).

Corollary 2. *If factoring integers* $n = p^2q$ *is hard, the map* $f_{pq} : \mathbb{Z}_{pq}^\times \to \text{N-R}(n)$, $x \mapsto x^n \bmod n$ *is a trapdoor one-way function.*

Proof. See Appendix A.1.

We now turn to the group $\mathbb{Z}_{n^2}^\times$ and analyze a Paillier-like function.

Theorem 2. *Let* $n \neq 0 \bmod 3$.
The map $f : \mathbb{Z}_n^\times \times \mathbb{Z}_n \to \mathbb{Z}_{n^2}^\times, (r, m) \mapsto r^n(1 + mn) \bmod n^2$ *has the following properties:*

1. f *is well-defined,* i.e. *if* $r = r' \bmod n$ *and* $m = m' \bmod n$ *holds, then it follows that* $r^n(1 + mn) = r'^n(1 + m'n) \bmod n^2$ *is true.*
2. f *is homomorphic in* r *and* m*,* i.e. $f(r_1r_2, m_1 + m_2) = f(r_1, m_1)f(r_2, m_2)$.
3. $f(r, m) = f(r + ipq, m - ir^{-1}pq)$ *for* $i \in \mathbb{Z}$*, hence* f *is* p*-to-one.*
4. $\text{im}(f) = \{x = x_0 + nx_1 \in \mathbb{Z}_{n^2}^\times | x_0 \in \text{N-R}(n), x_1 \in \mathbb{Z}_n\}$.
5. *The restrictions* $f_m = f|_{\mathbb{Z}_n^\times \times \mathbb{Z}_{pq}}$ *and* $f_r = f|_{\mathbb{Z}_{pq}^\times \times \mathbb{Z}_n}$ *are one-to-one.*
6. $f_r : \mathbb{Z}_{pq}^\times \times \mathbb{Z}_n \to \mathbb{Z}_{n^2}^\times$ *is a group homomorphism with respect to the group operation* $\circ_r$*:*
$$(r_1, m_1) \circ_r (r_2, m_2) = (\underbrace{r_1r_2 \bmod pq}_{=:r_{pq}}, m_1 + m_2 + lr_{pq}^{-1}pq \bmod n),$$
where $0 \leq l < p$ *is defined via* $r_1r_2 = r_{pq} + lpq \bmod n$.
7. $f_m : \mathbb{Z}_n^\times \times \mathbb{Z}_{pq} \to \mathbb{Z}_{n^2}^\times$ *is a group homomorphism with respect to the group operation* $\circ_m$*:*
$$(r_1, m_1) \circ_m (r_2, m_2) = (r_1r_2 - lpq \bmod n, \underbrace{m_1 + m_2 \bmod pq}_{=:m_{pq}}),$$
where $0 \leq l < p$ *is defined via* $m_1 + m_2 = m_{pq} - r_{pq}^{-1}lpq \bmod n$.

Proof. See Appendix A.2.

3 Homomorphic Encryption

In this section, we propose a new homomorphic encryption scheme based on the one-way function analyzed in Sect. 2.2. At Eurocrypt 1999, Pascal Paillier introduced a homomorphic encryption scheme that is IND-CPA secure in the standard model under an appropriate decisional assumption [Pai99]. Unfortunately, no proof is known that reduces the one-wayness of this scheme to a standard computational assumption[2]. In the following, several variants of Paillier's original scheme have been described, *e.g.* RSA-Paillier [CGHGN01], that significantly reduces the encryption costs and that is one-way under the standard RSA-assumption. However, the one-way reduction is not tight and the scheme is not homomorphic any more.

Next, we show that by replacing the modulus $n = pq$ with $n = p^2q$ in Paillier's scheme one obtains an encryption scheme with the following properties:

– The proposed scheme is one-way under a weak standard assumption (namely factoring $n = p^2q$).
– The security reduction is tight.
– The proposed scheme is IND-CPA under the same security assumption as Paillier's original one.
– The proposed scheme is still homomorphic.
– The only drawback of the new scheme is that it is vulnerable to a simple chosen ciphertext attack.

3.1 The Proposed Encryption Scheme

Key Generation: Let k be a security parameter. Choose two primes p,q of the same length such that both $p-1$ and $q-1$ have a large prime factor, $p \nmid q-1, q \nmid p-1$, and the product $n = p^2q$ is a k bit number. Compute $d = n^{-1} \bmod (p-1)(q-1)$ and let l be chosen such that $2^l < pq < 2^{l+1}$. The public key is $\mathrm{pk} = (n, l)$, and the secret key is $\mathrm{sk} = (d, p, q)$.

Encryption: To encrypt a message $m \in \{0,1\}^l$, choose $r \in \mathbb{Z}_n^\times$ at random and compute

$$\mathcal{E}_{\mathrm{pk}}(m, r) = r^n(1 + mn) \bmod n^2$$

Decryption: To decrypt a ciphertext $c \in \mathbb{Z}_{n^2}^\times$, first compute $r = c^d \bmod pq$. Then

$$\mathcal{D}_{\mathrm{sk}}(c) = L_n(r^{-n}c \bmod n^2) \bmod pq,$$

where the L-function is defined as $L_n(x) = \frac{x-1}{n}$.

We can prove the following theorem:

Theorem 3. *1. The proposed encryption scheme is correct.*
2. The proposed encryption scheme is homomorphic, i.e. *we have*

$$\mathcal{E}_{\mathrm{pk}}(r_1, m_1)\mathcal{E}_{\mathrm{pk}}(r_2, m_2) = \mathcal{E}_{\mathrm{pk}}((r_1, m_1) \circ (r_2, m_2)),$$

where $\circ$ is the group operation on $\mathbb{Z}_n^\times \times \mathbb{Z}_{pq}$ as defined in Theorem 2(7).
3. The proposed encryption scheme is one-way if factoring integers $n = p^2q$ is hard.

Proof. See Appendix A.3.

[2] The composite residuosity assumption, under which Paillier proved his scheme to be one-way, actually is just a paraphrase of the one-wayness property.

Remark 1. Although the group operation that makes the scheme homomorphic is non-standard, we want to point out that for almost all applications the following relaxed homomorphic property of the proposed scheme is sufficient: $\mathcal{E}_{\mathrm{pk}}(r_1, m_1)\mathcal{E}_{\mathrm{pk}}(r_2, m_2) = \mathcal{E}_{\mathrm{pk}}(r, m_1 + m_2)$ for some $r \in \mathbb{Z}_n^\times$.

We now show that the proposed scheme is semantically secure against passive adversaries under almost the same assumption as Paillier's original scheme. First, we review this assumption:

Definition 3 (Decisional Composite Residuosity Assumption). *Let $n = p^2q$ for two large primes p, q with $p \nmid q-1, q \nmid p-1$. Then the* Decisional Composite Residuosity Assumption *(DCRA) states that for any polynomial time distinguisher $\mathcal{A}^{DCRA}$ the following quantity (called $\mathcal{A}^{DCRA}$'s* advantage*) is negligible in* $\log(n)$*:*

$$|\Pr[x \leftarrow \mathbb{Z}_{n^2}^\times : \mathcal{A}^{DCRA}(x) = 1] - \Pr[x \leftarrow \mathbb{Z}_n^\times : \mathcal{A}^{DCRA}(x^n \bmod n^2) = 1]|.$$

The only difference to Paillier's Decisional Composite Residuosity Assumption is the choice of $n = p^2q$ instead of $n = pq$. As long as factoring integers of p^2q type is supposed to be as hard as factoring integers of pq type, there is no reason to believe that the tractability of DCR problem depends on the type of the modulus.

Semantic Security is defined as follows:

Definition 4 (Semantic Security). *A cryptosystem with key generator $\mathcal{G}$ and randomized encryption function $\mathcal{E}$ is semantically secure against passive adversaries (IND-CPA) if for any PPA $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following quantity (called $\mathcal{A}$'s* advantage*) is negligible in the security parameter k:*

$$|2\Pr[(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathcal{G}(k), \mathcal{A}_1(\mathrm{pk}) = (m_0, m_1, st), b \leftarrow \{0,1\}, c \leftarrow \mathcal{E}(pk, m_b) : \\ \mathcal{A}_2(m_0, m_1, c, st) = b] - 1|$$

Theorem 4. *The proposed scheme is IND-CPA if and only if the Decisional Composite Residuosity Assumption holds.*

Proof. See Appendix A.4.

4 Trapdoor Commitments in Digital Signature Schemes

In this section, we develop new trapdoor commitment schemes as applications of the one-way functions introduced in Sect. 2.2. Commitment schemes are an important primitive in public key cryptography. Among these, *trapdoor* commitment schemes play a leading role, especially in the context of zero-knowledge protocols.

Recently, trapdoor commitment schemes found new applications in a different setting, namely in designing digital signature schemes with additional properties, *e.g.* on-line/off-line signatures, chameleon signatures and signcryption [ST01, KR00, ADR02].

In this new context, trapdoor commitment schemes are not longer described as a protocol between the commiter and the receiver, but a functional oriented definition is more convenient, and the term trapdoor commitment scheme is replaced by trapdoor hash family, *aka* chameleon hash family. As pointed out by Krawczyk and Rabin [KR00], both definitions are equivalent (for *non-interactive* trapdoor commitment schemes).

Definition 5 (Trapdoor Hash Family). *A trapdoor hash family is a pair* $(\mathsf{KeyGen}, \mathsf{H})$ *of key generation algorithm and a family of polynomial time hash functions such that the following holds:*

- ***Key Generation:*** *On input a security parameter k, the randomized algorithm* KeyGen *outputs a pair* $(\mathrm{hk}, \mathrm{tk})$ *of hash and trapdoor key, where the sizes of* hk *and* tk *are polynomially related to k. The hash key* hk *uniquely specifies an element* $\mathrm{hash}_{\mathrm{hk}}$ *of the family* H.
- ***Hash:*** *The algorithm* $\mathrm{hash}_{\mathrm{hk}} : \mathcal{R} \times \mathcal{M} \to \mathcal{H}$ *computes the hash value, where* $\mathcal{M}$ *and* $\mathcal{R}$ *are the message space and the space of randomness.*
- ***Weak Altering:*** *For each trapdoor key* tk *there is a polynomial time algorithm* $w\,\mathrm{Alt}_{\mathrm{tk}} : \mathcal{M} \times \mathcal{R} \times \mathcal{M} \to \mathcal{R}$ *that given a message m, randomness r, and a target message m' computes randomness r' with* $\mathrm{hash}_{\mathrm{hk}}(r, m) = \mathrm{hash}_{\mathrm{hk}}(r', m')$. *The pair* $((r, m), (r', m'))$ *is called a* trapdoor collision.

The algorithm $w\,\mathrm{Alt}_{\mathrm{tk}}$ enables the owner of the trapdoor key to compute a hash value $\mathrm{hash}_{\mathrm{hk}}(r, m) = h$ and "alter" the meaning of h in any desired way (*i.e.* he can claim that h is the hash value of an arbitrary message $m' \in \mathcal{M}$ by presenting the randomness r' with $\mathrm{hash}_{\mathrm{hk}}(r', m') = h$ as a proof). However, for some applications a strictly stronger property turns out to be useful, namely the owner of the trapdoor key should be able to alter a hash value arbitrarily even without knowledge of the pre-image values r, m. We call this property *strong altering*[3]:

Definition 6 (Strong Altering). *A trapdoor hash family* $(\mathsf{KeyGen}, \mathsf{H})$ *provides strong altering, if for each key* $(\mathrm{hk}, \mathrm{tk}) \leftarrow \mathsf{KeyGen}(k)$ *there is a polynomial time algorithm* $s\,\mathrm{Alt}_{\mathrm{tk}} : \mathcal{H} \times \mathcal{M} \to \mathcal{R}$ *with the following property: Given a hash value* $h \in \mathcal{H}$ *and a target message* $m \in \mathcal{M}$, *the algorithm* $s\,\mathrm{Alt}_{\mathrm{tk}}$ *computes randomness* $r \in \mathcal{R}$ *with* $h = \mathrm{hash}_{\mathrm{hk}}(r, m)$.

The security of a trapdoor hash family requires that without knowledge of the trapdoor key it should be hard to find collisions. Moreover, the hash pairs obtained by invoking the altering algorithm should be indistinguishable from "real" hash pairs.

Definition 7 (Security of Trapdoor Hash Families). *The trapdoor hash family* $(\mathsf{KeyGen}, \mathsf{H})$ *is secure if the following properties hold:*

[3] In [ST01], this property is referred to as *inversion property*.

Collision resistance: *For any PPA $\mathcal{A}$ the following probability is negligible in k:*

$$\Pr[(\mathrm{hk}, \mathrm{tk}) \leftarrow \mathsf{KeyGen}(k), \mathcal{A}(\mathrm{hk}) = (r, m, r', m') : \\ (r, m) \neq (r', m'), \mathrm{hash}_{\mathrm{hk}}(r, m) = \mathrm{hash}_{\mathrm{hk}}(r', m')]$$

Uniformity: *The outcome of $s/w\,\mathrm{Alt}_{\mathrm{tk}}$ is uniformly distributed in $\mathcal{R}$ provided that the randomness input is also uniformly distributed in $\mathcal{R}$.*

4.1 Recent Applications for Trapdoor Hash Families

On-line/Off-line Signatures. A digital signature scheme possesses the *on-line/off-line property* if the signer is able to perform the bulk of the computation off-line, *i.e.* before receiving the message that has to be signed. In [ST01], Shamir and Tauman proposed a generic construction for transforming a weak secure signature scheme into a strong on-line/off-line signature scheme improving an early proposal of Even, Goldreich and Micali [EGM96]. Informally, their idea is the following: The signer runs two key generation algorithms, for an ordinary signature scheme and for a trapdoor hash family (in particular, the signer holds the hash trapdoor). In the off-line phase, the signer randomly selects r' and m' and constructs the dummy hash value $h = \mathrm{hash}_{\mathrm{hk}}(r', m')$. She signs h and keeps the obtained signature sig and the values r', m', h in memory. In the on-line phase, when the message m that has to be signed is known, the signer invokes the altering algorithm to obtain randomness r with $\mathrm{hash}_{\mathrm{hk}}(r, m) = h$. The tuple (r, sig) then defines the signature of m. Signature verification is straight-forward, as by construction sig is a valid hash-then-sign signature of m. Note that only for executing the protocol weak altering is sufficient. The strong altering property, however, leads to more powerful constructions: instantiated with a signature scheme that is existentially unforgeable under a generic chosen message attacks, the above construction (with a weak hash family) is secure against adaptive chosen message attacks (EF-CMA). If strong altering is possible, the same result can be proven for weaker input signature schemes (only security against random message attacks is required). To sum up, we are looking for a trapdoor hash family where strong altering is possible and weak altering is fast. Some previous constructions of trapdoor hash families provide strong altering algorithms, but the weak altering is not particularly efficient [Gen04, FF02, KR00]. In contrast, Shamir and Tauman constructed a new trapdoor hash family with a very fast weak altering mechanism, but lacking the strong altering property. In the following, we will give the first trapdoor hash family that combines both properties: extremely fast weak altering and the possibility of strong altering.

Signcryption. In [ADR02] An *et al.* give a formal treatment of securely carrying out joint encryption and signature in the public-key setting. Roughly speaking, the security goals here are to protect the sender's authenticity and the receiver's privacy. In addition to examining the classical "encrypt-then-sign" and "sign-then-encrypt" paradigms, An *et al.* propose a new variant which they call "commit-then-encrypt-and-sign". In this setting, it is possible to perform the costly public-key operations (encrypt and sign) in parallel, therefore saving computation time[4]. For the sake of uniformity, we describe the basic procedure in hash-terminology: The sender hashes the message, then in parallel he signs the hash and encrypts the message. Finally he transmits the hash data, the signature and the ciphertext to the receiver. Until now, the trapdoor has not been exploited. The benefit of a trapdoor hash family arises if for signing Shamir-Tauman on-line/off-line signatures are used. Indeed, combined with an on-line/off-line encryption scheme (*e.g.* hybrid encryption), a complete on-line/off-line signcryption scheme can be obtained as follows: In the off-line phase, the sender creates a fake hash $h(r', m')$, builds the signature sig of $h(r', m')$ and performs the off-line encryption operations (*e.g.* constructing and encapsulating a session key). Once the message m is known, the sender invokes the altering algorithm to obtain randomness r' with $h(r', m') = h(r, m)$, completes the encryption (*e.g.* encrypting m symmetrically with the session key) and finally transmits r, sig and the obtained ciphertext c. The receiver encrypts c, gets m and verifies if sig is a valid signature of $h(r, m)$. It has not been pointed out by An *et al.* that this procedure even enhances the security of the original signature scheme. But along the lines of Shamir-Tauman's security proof it can be easily shown that a signature scheme secure against generic message attacks is sufficient to receive an EF-CMA secure conversion. If the trapdoor hash family provides strong altering, even security against random message attacks suffices.

Chameleon Signatures. The concept of chameleon signatures was introduced by Krawczyk and Rabin to simplify undeniable signatures [KR00]. The aim of chameleon signatures is to distract the receiver of a signature from revealing the signed message to any third party. To realize this, the well-known hash-then-sign paradigm is used, but the conventional hash function is replaced by a trapdoor hash. Here, only the receiver holds the secret trapdoor and is therefore able to forge valid signatures for messages of his choice by invoking the altering algorithm. Consequently, no third party will be convinced of the validity of any signature, because the receiver could have created it himself. On the other hand, the signer is protected against an unhonest receiver who creates a fake valid signature, because in this case the legitimate signer is able to present a collision of the trapdoor hash function with overwhelming probability. This enables the signer to deny the forged signature. In [CGHGN01], Catalano *et al.* pointed out that on-line/off-line chameleon signatures can be constructed when using an on-line/off-line trapdoor hash family.

4.2 Our New Trapdoor Hash Families

The Proposed Strong Trapdoor Hash Family with Fast Altering. The following trapdoor hash family with fast altering can be compared to Shamir-

[4] Note that the naive approach of just signing and encrypting a message in parallel and sending the signature along with the ciphertext may violate the receiver's privacy, because the signature may reveal information about the message.

Tauman's fast altering trapdoor hash family from [ST01]. Shamir and Tauman use the homomorphic one-way function $x \mapsto g^x \bmod n$, where n is a safe RSA modulus and g is an element of maximal order $\lambda(n)$ in $\mathbb{Z}_n^\times$. We will use the one-way homomorphism $x \mapsto x^n \bmod n$ for $n = p^2q$. In both cases, the idea is that a non-trivial element of the function's kernel reveals the factorization of n (by providing a multiple of $\lambda(n)$, resp. pq), which implies collision resistance. The main difference is that only in our case the function is trapdoor, which provides the strong altering property.

Key Generation: Let k be a security parameter. Choose two primes p,q of the same length such that both $p-1$ and $q-1$ have a large prime factor, $p \nmid q-1, q \nmid p-1$, and the product $n = p^2q$ is a k bit number. Define l such that $2^l < pq < 2^{l+1}$ holds.
The hash key is $\mathrm{hk} = (n, l)$, and the trapdoor key is $\mathrm{tk} = (p, q)$.

Hash: To hash a message $m \in \{0, \ldots, [n]^{k-l-1} - 1\}$, a value $r \in \mathbb{Z}_{pq}$ is chosen uniformly at random and $\mathrm{hash}_{\mathrm{hk}}(r, m) = (m||r)^n \bmod n$ is computed, where $m||r$ denotes the concatenation of m and r.

In Appendix A.5, we prove the following theorem:

Theorem 5. *Under the p^2q factorization assumption the above construction is a secure trapdoor hash family with extremely fast weak altering and the strong altering property.*

Remark 2. In terminology of commitment schemes the above construction is a perfectly hiding and computationally binding trapdoor commitment scheme. We further want to point out that for the applications in on-line/off-line signature scheme resp. signcryption, the sender is also the trapdoor holder. Therefore it is not a problem for him to choose elements of $\mathbb{Z}_{pq}$ uniformly at random. For more general applications, the randomness has to be taken from the set $\{0, 1\}^l$ instead. However, in this case the proposed scheme is not longer perfectly binding. We assume that binding holds at least computationally.

The Proposed Strong Trapdoor Hash Family with On-Line/Off-Line Property.

Key Generation: Let k be a security parameter. Choose two primes p,q of the same length such that both $p-1$ and $q-1$ have a large prime factor, $p \nmid q-1, q \nmid p-1$, and the product $n = p^2q$ is a k bit number. Define l such that $2^l < pq < 2^{l+1}$ holds.
The hash key is $\mathrm{hk} = (n, l)$, and the trapdoor is the factorization of n.

Hash: To hash a message $m \in \{0, \ldots, [n]^{k-l-1} - 1\}$, one chooses two random values $s \in \mathbb{Z}_{pq}, r \in \mathbb{Z}_n^\times$ and computes $\mathrm{hash}_{\mathrm{hk}}(r, s, m) = (1 + (m||s)n)r^n \bmod n^2$, where $m||s$ denotes the concatenation of m and s.

In Appendix A.6, we prove the following theorem:

Theorem 6. *Under the p^2q factorization assumption the above construction is a secure trapdoor hash family with strong altering and on-line/off-line property.*

However, we still have to resolve a last problem: For sound execution the secret trapdoor has to be known to the user of the hash function, because he must choose s uniformly at random from $\mathbb{Z}_{pq}$. The same problem occurs in the factorization-based scheme from [BCP03], where the randomness has to be sampled from $\mathbb{Z}^{\times}_{n\lambda(n)/2}$. But unfortunately, in a chameleon hash signature scheme the trapdoor must be *unknown* to the signer/hasher, as otherwise the signer is able obtain hash collisions and thus to deny her signatures. This difficulty could be overcome by providing the user an upper bound of the secret numbers pq resp. $n\lambda(n)/2$, but is has to be pointed out that proceeding in that way usually leads to a loss of uniformity. But luckily, it turns out that in our case the problem can be solved nevertheless: if the randomness s is sampled from the publicly known set $\{0,1\}^{l+1}$ instead of $\mathbb{Z}_{pq}$, we can prove that the alleviated scheme fulfills the following relaxed uniformity requirement, which fortunately is enough for designing secure chameleon signature schemes (see [KR00]).

Definition 8 (Relaxed Uniformity for Trapdoor Hash Families). *For any PPA $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following is negligible in k:*

$$\begin{aligned}|2\Pr[(\mathrm{hk},\mathrm{tk}) \leftarrow \mathsf{KeyGen}(k), \mathcal{A}_1(\mathrm{hk}) = (m_0, m_1, st), b \leftarrow \{0,1\}, r \leftarrow \mathcal{R},\\ c \leftarrow \mathrm{hash}_{\mathrm{hk}}(r, m_b) : \mathcal{A}_2(m_0, m_1, c, st) = b] - 1|\end{aligned}$$

Lemma 1. *Relaxed Uniformity holds for the alleviated scheme under the decisional composite residuosity assumption.*

Proof. For any two messages m_0, m_1, the distributions of $\mathrm{hash}_{\mathrm{hk}}(r, s, m_0)$ and $\mathrm{hash}_{\mathrm{hk}}(r, s, m_1)$ are computationally indistinguishable if $s \in \{0,1\}^{l+1}, r \in \mathbb{Z}_n^{\times}$ are chosen uniformly at random. This is an immediate consequence of the semantic security of the encryption scheme described in Sect. 3 (Theorem 4). □

Remark 3. In terminology of commitment schemes the above construction is a perfectly hiding and computationally binding trapdoor commitment scheme. For the alleviated scheme both hiding and binding holds computationally.

4.3 Comparison with Previous Work

In this section, we compare previously known trapdoor hash families with our new constructions. In Table 1, we focus on schemes that are intended to be used in Shamir-Tauman on-line/off-line signature schemes. Here, the most important property is efficient weak altering. Furthermore, schemes allowing strong altering are preferable because they lead to more powerful conversions. In the last column, we note if the secret trapdoor has to be known to the sender. This is no problem in the suggested applications (on-line/off-line signatures, on-line/off-line signcryption), but for more general applications those schemes might be improper.

In Table 2, we compare different constructions for on-line/off-line trapdoor hash families. Here, it is notable that the need for the user to know the secret trapdoor excludes the application chameleon signatures, therefore we designate

Table 1. Comparison of trapdoor hash families suitable for [ST01]

Scheme	Assumption	strong	hash	weak alt.	user needs tk
[BK90]	Discrete log	NO	$\approx$ 1 exp.	$\approx$ 1 mult.	NO
[KR00]	Factoring	**YES**	$\approx \lvert m\rvert_2$ mult.	$\approx$ 5 mult.	NO
[ST01]	Factoring	NO	1 exp.	**1 add. + bit shift**	YES
[BCP03]	Factoring	NO	$\approx$ 1 exp.	$\approx$ 1 mult.	YES
1. proposed	Factoring	**YES**	1 exp.	**1 add. + bit shift**	YES

Table 2. Comparison of on-line/off-line trapdoor hash families

Scheme	Assumption	strong	hash	user needs tk
[CGHGN01]	RSA(n, n)	YES	$\approx$ 1 exp.	**NO**
[BCP03]	**Factoring**	NO	$\approx$ 1 exp.	YES
2. proposed	**Factoring**	YES	$\approx$ 1 exp.	**NO**

the alleviated version of our second scheme (where the randomness is sampled from $\{0, 1\}^{l+1}$).

In both tables, the assumption factoring refers to integers of p^2q-type for the proposed schemes and to RSA moduli, resp. Blum integers [KR00], else.

5 Conclusion

In this paper we pointed out that combining Paillier's homomorphic encryption scheme with the Okamoto-Uchiyama modulus $n = p^2q$ yields theoretical benefits as well as practical applications. First we developed a semantically secure, additively homomorphic variant of Paillier's encryption scheme that is one-way under the factoring assumption. We based the one-wayness proof on the analysis of new trapdoor one-way functions, which may be of independent interest. As a practical application, we constructed two new factorization-based trapdoor hash families. The first one provides as fast weak altering as Shamir-Tauman's proposal, but in addition allows strong altering too, yet leading to more powerful generic constructions of on-line/off-line signature schemes. The second one is the first factorization-based trapdoor hash family that enables the construction of on-line/off-line chameleon signatures (a previous construction exists based on the hardness of inverting RSA(n, n)).

Acknowledgements

The authors wish to thank anonymous referees and Benoit Chevallier-Mames for useful comments.

References

[ADR02] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[AM94] L. M. Adleman and K. S. McCurley. Open problems in number theoretic complexity, ii. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, pages 291–322. Springer, 1994.

[BCP03] E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2003.

[BDHG99] D. Boneh, G. Durfee, and N. Howgrave-Graham. Factoring $N = p^r q$ for large r. In Michael J. Wiener, editor, *Advances in Cryptology – Proceedings of CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 326–337, Berlin, 1999. Springer-Verlag.

[BK90] J. F. Boyar and S. A. Kurtz. A discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology*, 2(2):63–76, 1990.

[CF85] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Symposium on Foundations of Computer Science – Proceedings of FOCS 1986*, pages 372–382, 1985.

[CGHGN01] D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's cryptosystem revisited. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS-2001)*, pages 206–214, 2001.

[EGM96] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.

[FF02] M. Fischlin and R. Fischlin. The representation problem based on factoring. In Bart Preneel, editor, *CT-RSA*, volume 2271 of *Lecture Notes in Computer Science*, pages 96–113. Springer, 2002.

[FKM$^+$] E. Fujisaki, T. Kobayashi, H. Morita, H. Oguro, T. Okamoto, S. Okazaki, D. Pointcheval, and S. Uchiyama. EPOC: Efficient probabilistic public-key encryption. Submitted to ISO and NESSIE.

[FOM91] A. Fujioka, T. Okamoto, and S. Miyaguchi. ESIGN: An efficient digital signature implementation for smart cards. In Donald W. Davies, editor, *Advances in Cryptology – Proceedings of EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 446–457, Berlin, 1991. Springer-Verlag.

[Gen04] R. Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2004.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[KR00] H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS*. The Internet Society, 2000.

[Len87] H.W. Lenstra, Jr.. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.

[LL93] A.K. Lenstra and H.W. Lenstra, Jr., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, 1993.

[NS98] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS-1998)*, pages 59–66, New York, 1998. ACM Press.

[OP00] T. Okamoto and D. Pointcheval. EPOC-3 - efficient probabilistic public-key encryption, 2000. Submitted to IEEE P1363.

[OU98] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – Proceedings of EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 308–318. Springer-Verlag, 1998.

[Pai99] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – Proceedings of EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223 – 238. Springer-Verlag, 1999.

[PO96] R. Peralta and E. Okamoto. Faster factoring of integers of a special form. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, E79-A(4):489–493, 1996.

[ST01] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2001.

[Tak98] T. Takagi. Fast RSA-type cryptosystem modulo p^kq. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 318–326. Springer, 1998.

[Tak04] T. Takagi. A fast RSA-type public-key primitive modulo p^kq using Hensel lifting. *IEICE Transactions*, Vol.E87-A(1):94–101, 2004.

A Some Proofs

A.1 Proof of Corollary 2

Define $d = n^{-1} \bmod \varphi(pq)$ (note that $\gcd(n, \varphi(pq)) = 1$). Then d can be used as a trapdoor to invert f_{pq}, because the pre-image of $y \in$ N-R(n) is computed as $f_{pq}^{-1}(y) = y^d \bmod pq$. The one-wayness is a consequence of Theorem 1: To factor n with access to an oracle that inverts f_{pq}, we choose an element $x \in \mathbb{Z}_n^\times$ at random and query the oracle on $x^n \bmod n$ (note that the distributions of $\{x^n \bmod n | x \in \mathbb{Z}_n^\times\}$ and $\{x^n \bmod n | x \in \mathbb{Z}_{pq}^\times\}$ are identical). With probability $1 - 1/p$ we have $x > pq$ and the oracle will answer $x' \in \mathbb{Z}_{pq}^\times$ with $x \sim x', x \neq x' \bmod n$. Thus $\gcd(x - x', n) = pq$ reveals the factorization of n. □

A.2 Proof of Theorem 2

1. Straightforward (use $(r + in)^n = \sum_{j=0}^{n} \binom{n}{j} r^{n-j}(in)^j = r^n \bmod n^2$).
2. Straightforward computation.

3. We have

$$\begin{aligned}
f(r+ipq, m-ir^{-1}pq) &= (r+ipq)^n(1+(m-ir^{-1}pq)n)\\
&= \left(\sum_{j=0}^{3}\binom{n}{j}r^{n-j}(ipq)^j\right)(1+(m-ir^{-1}pq)n)\\
&= \left(r^n + nr^{n-1}ipq + \frac{n(n-1)}{2}r^{n-2}(ipq)^2 + \frac{n(n-1)(n-2)}{6}r^{n-3}(ipq)^3\right)\\
&\quad (1+(m-ir^{-1}pq)n)\\
&= (r^n + nr^{n-1}ipq)(1+(m-ir^{-1}pq)n)\\
&= r^n + n(r^{n-1}ipq + r^n m - r^n ir^{-1}pq) = r^n(1+mn) = f(r,m) \bmod n^2
\end{aligned}$$

Note that the second step is true, because $n^2 = p^4q^2$ is an integer divisor of $(ipq)^j$ for $j > 3$. The fourth step holds because n divides $n(n-1)/2$ and $\frac{n(n-1)(n-2)}{6}$ as n is odd and either $n-1$ or $n-2$ is a multiple of 3.

4. - 7. are more or less immediate consequences of 3. and 2.

□

A.3 Proof of Theorem 3

1. See Theorem 2(5).
2. See Theorem 2(7).
3. Let $\mathcal{O}_{OW}$ be an oracle that answers $m \in \{0,1\}^l$ on the input $c = r^n(1+mn) \bmod n^2$ for a $r \in \mathbb{Z}_n^\times$ with a non-negligible advantage ε. We show how to factor n with access to that oracle. First, we choose $m' \in \mathbb{Z}_n$ and $r \in \mathbb{Z}_n^\times$ at random and build the fake ciphertext $c' = r^n(1+m'n) \bmod n^2$. We distinguish two cases:

   Case 1: We have $|m' \bmod pq|_2 \le l$. From Theorem 2(3), we conclude that in this case the distribution of c' is exactly the same as the distribution of the original ciphertexts. Moreover, we conclude that $c' = \mathcal{E}_{\mathrm{pk}}(r', m' \bmod pq)$ for an appropriate $r' \in \mathbb{Z}_n^\times$ with $r = r' \bmod pq$. Thus, $\mathcal{O}_{OW}(c')$ will answer $m'_{pq} := m' \bmod pq$. With probability $1 - 1/p$, we have $m' > pq$ and thus we factor n via $\gcd(n, m' - m'_{pq}) = pq$.
   From the definition of l, we conclude that this case holds with probability greater than $1/2$.

   Case 2: We have $|m' \bmod pq|_2 > l$. In this case, c' is not an element of the regular ciphertext space and we cannot predict the oracle's behavior.

   Therefore, we factor the modulus with advantage at least $\frac{\varepsilon}{2}(1 - 1/p)$.

□

A.4 Proof of Theorem 4

To prove that the proposed scheme is IND-CPA under the DCRA, we construct a distinguisher $\mathcal{D}$ that breaks DCRA using the adversary $\mathcal{A}^{SS}$ against the semantic

security of the proposed scheme as a subroutine. Let x be an instance of the DCR problem. First, $\mathcal{D}$ runs $\mathcal{A}_1^{SS}$ on the pubic key and obtains two different messages $m_0, m_1 \in \mathbb{Z}_{pq}^{\times}$. Then, $\mathcal{D}$ chooses a bit $b \in \{0,1\}$ at random, computes $c = x(1+m_b n) \bmod n^2$, and runs $\mathcal{A}_2^{SS}$ on the triple (c, m_0, m_1). $\mathcal{D}$ returns 1 (indicating that x is a n-th residue) if $\mathcal{A}_2^{SS}$ answers b, otherwise, $\mathcal{D}$ returns 0. If x is an n-th residue, than c is a valid cipher of m_0 (distributed as original ciphertexts), otherwise c is a random element of $\mathbb{Z}_{n^2}^{\times}$. Thus, the advantage of $\mathcal{D}$ equals $\varepsilon/2$, where ε is the advantage of $\mathcal{A}^{SS}$.
To prove the opposite direction, we sketch how a distinguisher $\mathcal{D}$ of DCRA with non-negligible advantage ε can be used to break the semantic security of the proposed scheme. The adversary $\mathcal{A}^{SS}$ works as follows: $\mathcal{A}_1^{SS}$ chooses two different messages $m_0, m_1 \in \mathbb{Z}_{pq}^{\times}$ and a bit $b \in \{0,1\}$ at random. Given the challenge $c \in \mathbb{Z}_{n^2}^{\times}$, $\mathcal{A}_2^{SS}$ queries $\mathcal{D}$ on $c(1-m_b n) \bmod n^2$ and returns b if the answer is 1, $1-b$ otherwise. With a similar argumentation as above we conclude that the advantage of $\mathcal{A}^{SS}$ equals $\varepsilon/2$. □

A.5 Proof of Theorem 5

1. Extremely fast weak altering: From Theorem 1, we conclude that $m'||r' = m||r \bmod pq$ is equivalent to $\text{hash}_{\text{hk}}(m', r') = \text{hash}_{\text{hk}}(m, r)$, thus $r = 2^{l+1}(m'-m) + r' \bmod pq$ yields the desired result. r can be computed extremely fast as multiplication with 2^{l+1} is just a bit shift operation.
2. Strong altering: Let h be a possible hash value and let m be the target message. From Corollary 2, we conclude that $r = h^{1/n} - 2^{l+1}m \bmod pq$ leads to $h = \text{hash}_{\text{hk}}(r, m) = (m||r)^n = (2^{l+1}m + r)^n \bmod n$.
3. Uniformity of altering: The considerations above show that for any hash value h and any message m there is a unique $r \in \mathbb{Z}_{pq}$ with $h = \text{hash}_{\text{hk}}(r, m)$. Consequently, uniformity holds for both altering algorithms.
4. Collision resistance under the p^2q factorization assumption: As $\text{hash}_{\text{hk}}(r, m) = \text{hash}_{\text{hk}}(r', m')$ is equivalent to $(2^{l+1}m + r)^n = (2^{l+1}m' + r')^n \bmod n$, we have $2^{l+1}m + r = 2^{l+1}m' + r' \bmod pq$ using Theorem 1. Due to the length-restrictions of m and r, it is impossible that this equality holds modulo n, thus we must have $\gcd(m||r - m'||r', n) = pq$.

□

A.6 Proof of Theorem 6

1. Strong altering: Let h be a possible hash value and let m be the target message. We show how to construct randomness $s \in \mathbb{Z}_{pq}, r \in \mathbb{Z}_n^{\times}$ with $h = \text{hash}_{\text{hk}}(r, s, m) = (1 + (m||s)n)r^n = (1 + n(2^{l+1}m + s))r^n \bmod n$. From Theorem 2(3), we know that there are $r_{pq} \in \mathbb{Z}_{pq}^{\times}, m_{pq} \in \mathbb{Z}_{pq}, 0 \le i < p$ with $h = (1 + (m_{pq} - ir_{pq}^{-1}pq)n)(r_{pq} + ipq)^n \bmod n^2$. The values r_{pq} and m_{pq} can be computed from h: $r_{pq} = h^{1/n} \bmod pq, m_{pq} = L_n(r_{pq}^{-n}h \bmod n^2) \bmod pq$. Thus, to achieve $h = \text{hash}_{\text{hk}}(s, r, m)$, we have to find $r \in \mathbb{Z}_n^{\times}, s \in \mathbb{Z}_{pq}, 0 \le i < p$ with

$$2^{l+1}m + s = m_{pq} - ir_{pq}^{-1}pq \bmod n \quad (1)$$

$$r = r_{pq} + ipq \bmod n \quad (2)$$

The first equation uniquely determines $s \bmod pq$ and i. From (2), one immediately computes $r \bmod n$.

2. Weak altering: It is immediate from the considerations above that the following procedure is correct: Given $m, m' \in \{0, \ldots, [n]^{k-l-1} - 1\}, r \in \mathbb{Z}_n^\times, s \in \mathbb{Z}_{pq}$ the weak altering algorithm computes $r' \in \mathbb{Z}_n^\times, s' \in \{0,1\}^{l+1}$ by solving the following system of modular equations:

$$2^{l+1}m + s = 2^{l+1}m' + s' - ir^{-1}pq \bmod n$$
$$r = r' + ipq \bmod n$$

3. Collision resistance under the p^2q factorization assumption: First note that $\mathrm{hash}_{\mathrm{hk}}(r, s, m) = \mathrm{hash}_{\mathrm{hk}}(r', s', m')$ is equivalent to $(1 + n(2^{l+1}m + s))r^n = (1 + n(2^{l+1}m' + s'))r'^n \bmod n^2$. From Theorem 2(3), we therefore conclude

$$2^{l+1}m + s = 2^{l+1}m' + s' - ijpq \bmod n \text{ for some } 0 \le i, j < p \text{ and}$$
$$r = r' + ipq \bmod n$$

From the length restrictions of m, m', s and s', we conclude that $i = 0$ is impossible. Thus we can factor n by computing $\gcd(r - r', n) = pq$.

4. Uniformity of altering: From the consideration about strong altering we conclude that for each message m and each hash value h there is exactly one $s \in \mathbb{Z}_{pq}$ and one $r \in \mathbb{Z}_n^\times$ satisfying $\mathrm{hash}_{\mathrm{hk}}(r, s, m) = h$. Thus the uniformity property trivially holds for both altering algorithms.
5. The proposed scheme possesses the on-line/off-line property: After the message m is retrieved, only one modular multiplication has to be performed, as the computation of $r^n n \bmod n^2$ can be done in advance.

□

B The Hardness of the p^2q Factoring Problem

Recently, the use of p^2q type moduli (or more general p^kq) attracted much attention in cryptography. For example, the modulus p^2q is used in the famous family of EPOC cryptosystems (based on Okamoto-Uchiyama homomorphic encryption) [FKM+, OU98, OP00] and in the signature scheme ESIGN [FOM91], whereas moduli p^kq can be utilized to enhance the decryption speed in RSA-type encryption schemes [Tak98, Tak04].

Numerous researchers tried to exploit the special form of those integers to find faster factorization methods [AM94, PO96, BDHG99]. But unless the exponent k in p^kq is not too large, the most efficient methods for factoring $n = p^kq$ are still Lenstra's elliptic curve method (ECM) [Len87], its improvements [PO96], and the number field sieve (NFS) [LL93]. More precisely, if the size of the smallest prime factor of n exceeds some bound (about 200 bits), the NFS is the method of choice. Consequently, if n is sufficiently large (*i.e.* 1024 bits), the special form $n = p^2q$ causes no problem, because in contrast to ECM the runtime of the NFS only depends on the size of n, not on the size of the smallest prime factor of n. Concluding, although it is not known if factoring $n = p^2q$ is more tractable than factoring $n = pq$ or not, the p^2q factorization assumption is well-investigated and therefore can be regarded as fairly weak.

Homomorphic Cryptosystems Based on Subgroup Membership Problems

Kristian Gjøsteen

Department of Telematics,
Norwegian University of Science and Technology, 7491 Trondheim, Norway
kristian.gjosteen@item.ntnu.no

Abstract. We define an abstract subgroup membership problem, and derive a number of general results for subgroup membership problems. We define an homomorphic public key cryptosystem based essentially on a subgroup membership problem, and show that this abstract construction gives a uniform description of many famous cryptosystems, such as ElGamal, Goldwasser-Micali and Paillier. We show that the abstract theory gives new insights into older results, and allows us to derive new results.

Keywords: public key encryption, homomorphic cryptosystems, subgroup membership problem.

1 Introduction

A cryptosystem is *homomorphic* with respect to some operation $*$ on the message space if there is a corresponding operation $*'$ on the ciphertext space such that for encryptions c, c' of messages m, m', $c *' c'$ is an encryption of $m * m'$.

Goldwasser and Micali [9] introduced the concept of semantic security and described a semantically secure cryptosystem based on the quadratic residue assumption. It was later remarked that this cryptosystem really was homomorphic with respect to addition in $\mathbb{Z}_2$.

ElGamal [8] introduced an homomorphic cryptosystem based on the Diffie-Hellman key exchange protocol [7]. It has later been remarked that this cryptosystem is semantically secure under the Decision Diffie-Hellman assumption.

Naccache and Stern [11] introduced an homomorphic cryptosystem based on "higher residues" in $\mathbb{Z}_n^*$, where n was a product of two primes of a special form. Okamoto and Uchiyama [13] described an homomorphic cryptosystem over $\mathbb{Z}_n^*$ using a modulus of the form $n = p^2q$. Paillier [14] introduced an homomorphic cryptosystem based on the ring $\mathbb{Z}_{n^2}$, where n was simply an RSA modulus.

It turns out that all of these cryptosystems are simply special cases of a cryptosystem based on a general subgroup membership problem [5]. We discuss some general theory for subgroup membership problems in Sect. 2, then we describe the cryptosystem in Sect. 3. We give a catalogue of known subgroup membership problems in Sect. 4, and describe some new results in Sect. 5.

The main achievement in this paper is an abstract construction for a homomorphic cryptosystem. From this construction, we derive as special cases many famous cryptosystems. The use of abstract descriptions to focus on the interesting points of a cryptosystem is a common technique, see for example [3]. The novel arguments in Sect. 2.1 and in Sect. 5 show that our abstract constructions gives us new insights into old results, and can reduce the analysis of new constructions to the analysis of older, simpler constructions.

2 Subgroup Membership Problems

A *subgroup membership problem* consists of a finite abelian group G along with a proper, non-trivial subgroup K. The problem is to decide if a group element $x \in G$ is in K or in $G \setminus K$. We denote this subgroup membership problem by $\mathcal{SM}_{(G,K)}$, and the advantage of an adversary A is

$$\mathrm{Adv}_A^{\mathcal{SM}_{(G,K)}} = |\Pr[A(G,K,x) = 1 \mid x \xleftarrow{r} K] - \Pr[A(G,K,x) = 1 \mid x \xleftarrow{r} G \setminus K]|.$$

When we leave out the adversary, we consider the maximal advantage of all algorithms using less than some fixed amount of resources.

An alternative description of the subgroup membership problem is that given a representative x of a residue class in the factor group G/K, the adversary must decide if this residue class is the neutral element in G/K. (Note that the residue class $xK = 1K$ if and only if $x \in K$.)

We note some general facts about subgroup membership problems.

Let $\mathcal{SM}_{(G,K)}$ be such that the factor group G/K is cyclic. If $|G/K|$ is a known prime ℓ, sampling a uniformly from $\{1, \dots, \ell - 1\}$ gives us a random automorphism $x \mapsto x^a$ on G/K.

If $|G/K|$ contains no small primes, then an element chosen uniformly at random from G/K is, except with negligible probability, a generator. Sampling a uniformly from $\{1, \dots, 2^N\}$ (for some sufficiently large N) therefore gives us, except with negligible probability, a representative x^a for a residue class in G/K chosen uniformly at random.

If we are given an element x, we sample a as above and x' uniformly from K to get $x^a x'$. If x is in K, then $x^a x'$ is an element of K chosen uniformly at random. If x is in $G \setminus K$, then $x^a x'$ is (except with at most negligible probability) a random representative of a random non-neutral element in G/K, that is, $x^a x'$ is a random element in $G \setminus K$.

Therefore, $\mathcal{SM}_{(G,K)}$ is random self-reducible.

Note the following useful extension of this idea. Let $\mathcal{SM}_{(G,K)}$ and $\mathcal{SM}_{(G',K')}$ be subgroup membership problems. If there is a probabilistic algorithm that on input of x sampled uniformly from K outputs an element of K', and on input of x sampled uniformly from $G \setminus K$ outputs an element of $G' \setminus K$, with the output distribution in both cases uniform, then

$$\mathrm{Adv}^{\mathcal{SM}_{(G',K')}} \leq \mathrm{Adv}^{\mathcal{SM}_{(G,K)}}.$$

The following theorem will be useful for the analysis in Sect. 4 and 5.

Theorem 1. *Let $\mathcal{SM}_{(G,K)}$ be a subgroup membership problem, and let K' be a proper, non-trivial subgroup of K. Then*

$$\mathrm{Adv}^{\mathcal{SM}_{(G,K')}} \leq \frac{|G|-|K|}{|G|-|K'|}\mathrm{Adv}^{\mathcal{SM}_{(G,K)}} + \frac{|G|(|K|-|K'|)}{|K|(|G|-|K'|)}\mathrm{Adv}^{\mathcal{SM}_{(K,K')}}.$$

Proof. Any adversary A against $\mathcal{SM}_{(G,K')}$ is also an adversary against $\mathcal{SM}_{(G,K)}$ and $\mathcal{SM}_{(K,K')}$. Define $wt(S) = \Pr[A(G,K,x)=1|x \leftarrow S]$. A tedious calculation shows that

$$\begin{aligned}&wt(G \setminus K') - wt(K') = \\ &\frac{|G|-|K|}{|G|-|K'|}(wt(G\setminus K) - wt(K)) + \frac{|G|(|K|-|K'|)}{(|G|-|K'|)|K|}(wt(K\setminus K') - wt(K')).\end{aligned}$$

Taking absolute values, the theorem follows. □

A problem related to the subgroup membership problem is the splitting problem. A *splitting problem* consists of a finite abelian group G along with two proper, non-trivial subgroups K and H with trivial intersection, and such that $G = KH$.

It is easy to show that the map from $K \times H$ to G given by $(x,y) \mapsto xy$ is a group isomorphism. Denote the inverse isomorphism by $\sigma : G \to K \times H$. The splitting problem is, given $z \in G$, compute $\sigma(z)$. We often say that x is the *projection of z on K*, and y is the *projection of z on H*. We denote the splitting problem by $\mathcal{SP}_{(G,K,H)}$.

If $\gcd(|K|,|H|) = 1$, anyone who knows $|K|$ and $|H|$ can compute σ by first computing $d \equiv |H|^{-1} \pmod{|K|}$ and then $\sigma(z) = (z^{d|H|}, z^{1-d|H|})$.

If algorithms are available for sampling K and H uniformly at random, the splitting problem is self-reducible, since the splitting of z can be recovered from the splitting of zxy, where $(x,y) \in K \times H$.

The splitting problem $\mathcal{SP}_{(G,K,H)}$ is hard if one of the subgroup membership problems $\mathcal{SM}_{(G,K)}$ and $\mathcal{SM}_{(G,H)}$ are hard. There exists group structures where the splitting problem is believed to be hard, while the subgroup membership problem is easy [10].

Let $\mathcal{SP}_{(G,K,H)}$ be a splitting problem such that H is cyclic, and let g be a group element such that its residue class generates G/K. Let $\lambda : G \to \mathbb{Z}_{|H|}$ be the group homomorphism defined by $\lambda(g) = 1$, and $\ker\lambda = K$. Sometimes, it will be convenient to regard λ as a map $G \to \mathbb{Z}$, with the requirement that $0 \leq \lambda(z) < |H|$ for all $z \in G$.

The *subgroup discrete logarithm problem* is then, given $z \in G$, compute $\lambda(z)$. We denote this problem by $\mathcal{SDL}_{(G,K,H,g)}$. An alternative point of view is to consider the subgroup discrete logarithm problem as the discrete logarithm problem in the factor group G/K. We shall return to that in Sect. 2.1.

If $g \in H$, the subgroup discrete logarithm problem is at least as hard as the splitting problem, since the splitting can be recovered from the subgroup discrete logarithm via $(zg^{-\lambda(z)}, g^{\lambda(z)})$.

For a given splitting problem $\mathcal{SP}_{(G,K,H)}$ with H cyclic, if there is some efficient algorithm for computing discrete logarithms in the subgroup H, the splitting problem is at least as hard as the subgroup discrete logarithm problem.

2.1 Complexity Theoretic Results

We discuss some complexity-theoretic results for the subgroup discrete logarithm problem. *To simplify the exposition, we shall simply assume that the problem instance $\mathcal{SDL}_{(G,K,H,g)}$ has been chosen according to a parameter τ, and when we say polynomial, we really mean polynomial in τ. The words significant and negligible have their usual meanings in this context and relate to τ.*

The main idea in this section is to observe that the subgroup discrete logarithm problem is the same as the discrete logarithm problem in the factor group G/K.

The main obstacle to computing discrete logarithms in the factor group, is that testing two residue class representatives for equality is equivalent to the subgroup membership problem. If there is an algorithm for testing residue classes for equality, we can apply several standard discrete logartihm algorithms to the subgroup discrete logarithm problem.

We note that some discrete logarithm algorithms, such as Shank's Baby-step Giant-step, requires a canonical representative for the residue classes. These algorithms cannot be used.

Theorem 2. *Let $\mathcal{SDL}_{(G,K,H,g)}$ be a subgroup discrete logarithm problem such that $|H| = \ell^k$ for some prime ℓ, where ℓ is polynomial. Let H' be the unique subgroup of H of order ℓ. If there is a polynomial-time algorithm for solving the subgroup membership problem $\mathcal{SM}_{(KH',K)}$ with significant advantage, then there is a polynomial-time algorithm for solving $\mathcal{SDL}_{(G,K,H,g)}$ with significant advantage.*

Proof. The basic idea is that of ℓ-adic computation, as in the Pohlig-Hellman algorithm for computing discrete logarithms.

Let $z \in G$. Note that $z^{\ell^{k-1}} g^{-\ell^{k-1} m_0}$ is an instance of $\mathcal{SM}_{(KH',K)}$ for all integer m_0. By the self-reducibility of the subgroup membership problem, we can solve this problem with an advantage arbitrarily close to 1 (using a majority-vote algorithm). Therefore, we can decide if $z^{\ell^{k-1}} g^{-\ell^{k-1} m_0} \in K = \ker \lambda$, for $m_0 = 0, \dots, \ell - 1$. This will give us m_0 such that $\lambda(z) \equiv m_0 \pmod{\ell}$

Now we know that zg^{-m_0} has order at most ℓ^{k-1} in G/K, so we test $z^{\ell^{k-2}} g^{-\ell^{k-2}(m_0 + m_1 \ell)}$ for $m_1 = 0, \dots, \ell - 1$. This gives us m_1 such that $\lambda(z) \equiv m_0 + m_1 \ell \pmod{\ell}$. In this way we recover $\lambda(z)$.

It is easy to show that parameters for the above algorithm can be selected to get a polynomial-time algorithm with significant advantage. □

We can also use Pollard's ρ-algorithm to compute discrete logarithms in the factor group. Combined with the above algorithm, we get a more efficient algorithm.

Next, we note that hard core bit results for the discrete logarithm problem also translate into hard core bit results for the subgroup discrete logarithm problems.

Theorem 3. *Let $\mathcal{SDL}_{(G,K,H,g)}$ be a subgroup discrete logarithm problem, let $d > 1$ be an integer relatively prime to $|H|$, and let D be a publicly known integer such that $d \equiv D^{-1} \pmod{|H|}$. If there is a polynomial-time algorithm for computing λ modulo d with a significant advantage, then there is a polynomial-time algorithm for solving $\mathcal{SDL}_{(G,K,H,g)}$ with significant advantage.*

This theorem was proved in [4] for the special case when $d = 2$. In the abstract setting, the standard proof that the least significant bit of the discrete logarithm problem is hard core suffices.

When $d > 2$, one must use a multinomial distribution instead of binomial, which somewhat complicates the proof and weakens the bounds, but the essential method is the same. We therefore omit the proof also in this case.

We also make the (novel) observation that any inverse to d is acceptable, so one does not need to know $|H|$ to apply this theorem, only a number n such that $|H|$ divides n and d does not. In this case, one must guess $|H|$ approximately which weakens the result somewhat.

Finally, we remark that polynomial-time algorithms for computing the ith d-adic digit of λ gives polynomial-time algorithms for computing λ when the answer is known to be less than $d^{\lfloor \log_d |H| \rfloor - i}$. For $d = 2$, it can be shown that this implies the simultaneous security of all the binary digits up to the ith. It is reasonable to look upon this as a limiting process, where the limit is the subgroup membership problem. There, it is hard to compute $\lambda(z)$ (ie. producing evidence for a given value of $\lambda(z)$), even when $\lambda(z) = 0$.

Unfortunately, the bounds in these bit security results are not sharp. Even if there are no algorithms for solving the underlying hard problem with a significant probability using a fixed amount of resources, there may be algorithms that compute the bits with significant probability using the same fixed amount of resources.

3 Cryptosystems

A public key cryptosystem consists of three algorithms: a key generation algorithm that outputs a public and a private key; an encryption algorithm that takes a public key and a message as input and outputs a ciphertext; and a decryption algorithm that takes a ciphertext and a private key as input and outputs a message. We require that encryptions of any message always decrypt to the same message.

The *real-or-random game* is played between a simulator and an adversary against the cryptosystem. First, the simulator generates a public key and gives it to the adversary. The adversary produces a single message. The simulator either encrypts this message, or a different message chosen uniformly at random. The adversary receives the resulting ciphertext, and must guess whether or not it decrypts to the message he chose.

A public key cryptosystem is said to be *semantically secure* if no computationally bounded adversary can learn anything about the decryption from a ciphertext that he did not already know before he saw the ciphertext. It can be shown that this notion is equivalent to saying that no computationally bounded adversary can win the real-or-random game with probability significantly different from 1/2.

A weaker notion of security is *one-way.* The adversary is given a public key and an encryption of a random message, and must recover that message.

Let $\mathcal{SP}_{(G,K,H)}$ be a splitting problem, and let $\mathcal{SM}_{(G,K)}$ be the corresponding subgroup membership problem. Let H' be a group isomorphic to H. The basic idea for this cryptosystem is that H should hold the message, and K should hold noise that conceals the message.

We need three maps, $\pi : G \to H$, $f : H' \to G$ and $f' : H \to G$. The map π is the projection on H: if $\sigma(z) = (x, y)$, then $\pi(z) = y$. This means that π is a group homomorphism. It is also an endomorphism on G and induces the identity map on the factor group G/K.

The map f' should be a group isomorphism. Note that H is isomorphic to the factor group G/K (the isomorphism is simply $x \mapsto xK$), so f' induces an isomorphism $\bar{f}' : G/K \to H'$.

The map f need not be a group homomorphism, but it induces a map $\bar{f} : H' \to G/K$. The induced map should be a group isomorphism, and it should be the inverse of $\bar{f}'$. Given a description of $\mathcal{SP}_{(G,K,H)}$ and H', it should be easy to find a description of f.

We describe a cryptosystem based on the group structure and corresponding maps. The key generation algorithm is a probabilistic polynomial-time algorithm that selects a group structure $\mathcal{SP}_{(G,K,H)}$, group H', along with a description of the splitting map $\sigma : G \to K \times H$, and maps $f : H' \to G$ and $f' : H \to H'$. The public key is $(\mathcal{SP}_{(G,K,H)}, f)$, the private key is $(\mathcal{SP}_{(G,K,H)}, f', \sigma)$.

The encryption algorithm takes as input the public key and a message $m \in H'$. It then uses the subgroup membership problem's sampling algorithms to sample a random element r of K and outputs the ciphertext $rf(m)$.

The decryption algorithm takes as input the private key and a ciphertext $c \in G$. It uses the splitting map to find $\sigma(c) = (x, y)$, and outputs the message $f'(y)$ (alternatively, $f'(\pi(c))$).

Consider the following diagrams, where $\mathcal{E}$ denotes encryption:

$$\begin{tikzcd}
G \arrow[dd, "\pi"'] & & & G/K \arrow[dd, "\mathrm{id}"'] & \\
 & H' \arrow[ul, "\mathcal{E}"'] & \Rightarrow & & H' \arrow[ul, "\bar{f}"'] \\
H \arrow[ur, "f'"'] & & & G/K \arrow[ur, "\bar{f}'"'] &
\end{tikzcd}$$

To see that encryptions of $m \in H'$ always decrypt to m, we look at the factor group. The encryption algorithm just computes the induced map $\bar{f} : H' \to G/K$, randomising the residue class representative. The map π is simply the identity on G/K, so $\pi \circ \bar{f} = \bar{f}$. Finally, $\bar{f}' : G/K \to H'$ is an isomorphism and the inverse of $\bar{f}$, so $\bar{f}' \circ \pi \circ \bar{f}$ is the identity on H'.

By the same argument, it is clear that the cryptosystem is homomorphic. The operation on messages is the group operation in H', the operation on ciphertexts is the group operation in G.

Theorem 4. *The above cryptosystem is semantically secure if and only if the subgroup membership problem $\mathcal{SM}_{(G,K)}$ is hard.*

Proof. We show that adversaries who win the real-or-random game can be turned into algorithms that distinguish K, and vice versa.

Suppose we have an adversary that attacks the cryptosystem, and that we are given a subgroup membership problem $\mathcal{SM}_{(G,K)}$ along with a group element z. We simulate the public key by finding (some) f and H' (which we can do by assumption) and give the resulting public key to the adversary. The adversary gives us a message m. We then give the adversary the ciphertext $zf(m)$. If $z \in K$, this will be an encryption of m. If $z \notin K$, then the ciphertext will be an encryption of a random message. Therefore, the adversary's answer gives us the answer to the subgroup membership problem, whenever the adversary is correct.

Second, suppose we have an adversary against the subgroup membership problem. Our adversary outputs the identity (in H') as the message, and receives a ciphertext z. If z decrypts to the identity, then $z \in K$, otherwise $z \notin K$, and in both cases, the distribution of z is the correct uniform distribution. The adversary against the subgroup membership problem then decides if z decrypts to the identity or not. □

Theorem 5. *If the map f in the above cryptosystem satisfies $f(H') = H$, then the cryptosystem is one-way if and only if the splitting problem $\mathcal{SP}_{(G,K,H)}$ is hard.*

Suppose we have a subgroup discrete logarithm problem $\mathcal{SDL}_{(G,K,H,g)}$, along with an algorithm for computing discrete logarithms in H. We can then specify an instantiation of the above cryptosystem. The message group H' is simply $\mathbb{Z}_{|H|}$. The map $f : \mathbb{Z}_{|H|} \to G$ is given by $f([m]) = g^m$, where the representative for the residue class $[m]$ is chosen according to some (arbitrary) rule. The map $f' : H \to \mathbb{Z}_{|H|}$ is simply the discrete logarithm map.

The following theorem holds.

Theorem 6. *The above cryptosystem instantiated as above with a subgroup discrete logarithm problem $\mathcal{SDL}_{(G,K,H,g)}$ is one-way if and only if the subgroup discrete logarithm problem is hard.*

Note that sometimes, $|H|$ is not known. This is not a problem.

4 Catalogue

We shall discuss several concrete subgroup membership problems, discuss the related splitting problems or subgroup discrete logarithm problems. When necessary, we also discuss the sampling algorithms, the splitting map σ, the generator g, the maps f, f' and the group H' required to instantiate the cryptosystem described in Sect. 3.

Except for a variant of the higher residue problem, all of the group structures described in this section have previously appeared in the literature.

There are other interesting subgroup membership problems, but since they do not fit the framework in Sect. 3, we do not include them.

Diffie-Hellman. Let L be a prime-ordered group with generator g, and let w be an integer relatively prime to the group order. Let $G = L \times L$. The subgroup K of G is generated by the element (g, g^w) The subgroup H is generated by $(1, g)$. The subgroup membership problem is called *Decision Diffie-Hellman*, the splitting problem is called *Computational Diffie-Hellman.*

The splitting map σ is given by $\sigma(x, y) = ((x, x^w), (1, yx^{-w}))$. The group H' is simply L, with the obvious maps f and f'. The resulting cryptosystem was proposed by ElGamal [8].

It is worth noting that the Diffie-Hellman problems have an additional self-reducibility property, in that the subgroup K can be changed. This means that all such subgroups are equally hard to distinguish. See [7,2] for further details.

Quadratic Residues. Let p and q be primes congruent to 3 modulo 4, and let $n = pq$. Let J_n be the subgroup of $\mathbb{Z}_n^*$ with Jacobi symbol 1, and let Q_n be the subgroup of quadratic residues of J_n. Then we have a subgroup membership problem $\mathcal{SM}_{(J_n,Q_n)}$ and a splitting problem $\mathcal{SP}_{(J_n,Q_n,\langle -1\rangle)}$. This is the *Quadratic Residue* problem, which first appeared in [9].

Since $\langle -1\rangle$ has order 2, the subgroup membership problem and the splitting problem are equivalent.

The splitting map can be computed using the Legendre symbol modulo either p or q. Since $H = \{1, -1\}$, we can either choose $H' = H$ or $H' = \mathbb{Z}_2$, with the obvious maps.

Higher Residues. Let p and q be primes congruent to 3 modulo 4, and let $n = pq$. Let Q_n be the subgroup of $\mathbb{Z}_n^*$ of quadratic residues. If c is an odd prime such that c divides $p-1$, but not $q-1$, we get, in an analogue of the quadratic residue problem, the *higher residue* problem, where H is a subgroup of Q_n of order c. This was first investigated by Cohen Benaloh for non-prime but smooth c ([1] and the references therein). We get a subgroup membership problem, a splitting problem and a natural subgroup discrete logarithm problem (discrete logarithms are computed using Pohlig-Hellman).

The natural generalisation is to let $p = 2ac + 1$, $q = 2bd + 1$ and $n = pq$, a, b prime, c, d smooth, such that $\gcd(ac, bd) = \gcd(ab, cd) = 1$. This was investigated by Naccache and Stern [11]. We let $G = Q_n$, K be the unique subgroup of order

ab and H be the subgroup of order cd. It is natural to assume that cd is publicly known.

Non-trivial elements of H immediately lead to a factorisation of the modulus n, so the element g should be a generator for Q_n.

Under the assumption that c and d are known (not only cd), Naccache and Stern [11] proposed an attack that required approximately

$$O(n^{1/2}/(cd)^2) \tag{1}$$

work. This means that if cd is too large, there are better algorithms for factoring the modulus than the general factoring algorithms.

The basic idea of the attack is as follows. We have that $n = 4abcd + 2(ac + bd) + 1$. Then we get that

$$n - 1 \equiv 2ac \pmod{d} \text{ and } n - 1 \equiv 2bd \pmod{c}.$$

In other words, knowledge of c and d gives us some knowledge about a and b that speeds up a search. If c and d are sufficiently small, factoring n seems to be the best attack on the subgroup membership problem $\mathcal{SM}_{(G,K)}$.

We sketch a variant of this problem in Appendix A.

Okamoto-Uchiyama. Let p and q be primes such that $\gcd(pq, \phi(pq)) = 1$, and let $n = p^2q$. It is easy to see that $\mathbb{Z}_n^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^* \times \mathbb{Z}_p$, where $\mathbb{Z}_p$ is taken to be an additive group. The group G is $\mathbb{Z}_n^*$. The subgroup isomorphic to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ is K and the subgroup isomorphic to $\mathbb{Z}_p$ is H.

Considering the modulus p^2, we find that

$$(1 + p)^a \equiv 1 + \binom{a}{1}p + \binom{a}{2}p^2 \cdots \equiv 1 + ap \pmod{p^2},$$

so $(1 + p)$ has order p modulo p^2, and computing discrete logarithms in H is easy.

Sampling elements from K can be done by sampling uniformly from G, and then raising to the nth power.

Anyone who can solve the subgroup discrete logarithm problem, can recover the order of H, which is p, so he can factor n. It is not known if solving the subgroup membership problem is equivalent to factoring.

This group structure and the resulting cryptosystem was first proposed by Okamoto and Uchiyama [13].

Composite Residuosity. This subgroup membership problem was first proposed by Paillier [14], who improved on the previous work by Okamoto and Uchiyama. Our description also incorporates several later simplifications [6].

Let $n = pq$ be such that $\gcd(n, \phi(n)) = 1$. Let $G = \mathbb{Z}_{n^2}^*$. It is clear that $|G| = \phi(n)n$. Let K be the subgroup of order $\phi(n)$, and let H be the subgroup of order n.

As above,

$$(1 + n)^a \equiv 1 + \binom{a}{1}n + \binom{a}{2}n^2 \cdots \equiv 1 + an \pmod{n^2}.$$

From this, we see that discrete logarithms are easy to compute in H, and that $1+n$ generates H.

The subgroup membership problem $\mathcal{SM}_{(G,K)}$ and the subgroup discrete logarithm problem $\mathcal{SDL}_{(G,K,H,1+n)}$ are called the *Decision* and Computational Composite Residuosity problems.

Damgård and Jurik [6] describe a generalised group structure, using $G = \mathbb{Z}^*_{n^{s+1}}$ for some $s \geq 1$. As above, K is the subgroup of order $\phi(n)$ and H is the subgroup of order n^s. They get a subgroup membership problem $\mathcal{SM}_{(G,K)}$, and they show that it does not get much easier as s increases. We sketch the proof.

Let K_s and H_s be the subgroups of $\mathbb{Z}^*_{n^{s+1}}$ isomorphic to $\mathbb{Z}^*_n$ and $\mathbb{Z}_{n^s}$, respectively. We shall proceed by induction, using the following diagram:

$$\begin{tikzcd}
 & K_sH_s & \\
K_{s-1}H_{s-1} \arrow[ur] & K_sH_s^{n^{s-1}} \arrow[u, dash] & K_1H_1 \arrow[l] \\
K_{s-1} \arrow[u, dash] \arrow[ur] & K_s \arrow[u, dash] & K_1 \arrow[l] \arrow[u, dash]
\end{tikzcd}$$

It is easy to see that there exists a group homomorphism $K_1H_1 \to K_sH_s^{n^{s-1}}$ that takes K_1 into K_s. Therefore, any distinguisher for $\mathcal{SM}_{(K_sH_s^{n^{s-1}},K_s)}$ can be applied to $\mathcal{SM}_{(K_1H_1,K_1)}$.

Likewise, there exists an algorithm that takes $K_{s-1}H_{s-1}$ into K_sH_s such that K_{s-1} goes to $K_sH_s^{n^{s-1}}$. Again, this allows us to apply any distinguisher for $\mathcal{SM}_{(K_sH_s,K_sH_s^{n^{s-1}})}$ to $\mathcal{SM}_{(K_{s-1}H_{s-1},K_{s-1})}$.

Theorem 1 says that if we can solve $\mathcal{SM}_{(K_sH_s,K_s)}$, then we can solve either $\mathcal{SM}_{(K_sH_s,K_sH_s^{n^{s-1}})}$ or $\mathcal{SM}_{(K_sH_s^{n^{s-1}},K_s)}$.

Now we consider only distinguishing algorithms up to a certain fixed cost. Let ϵ_i be the maximal advantage of these algorithms against the subgroup membership problem $\mathcal{SM}_{(K_iH_i,K_i)}$. By Theorem 1 and the above discussion, we get that (ignoring some coefficients negligibly different from 1) $\epsilon_i \leq \epsilon_{i-1} + \epsilon_1$, or $\epsilon_s \leq (s-1)\epsilon_1$. In other words, the advantage increases at most linearly as s increases.

5 New Results

Let $p = 2ac+1$, $q = 2bd+1$ and $n = pq$, p, q, a and b prime, c and d smooth, such that $\gcd(ac,bd) = \gcd(ab,cd) = \gcd(n,\phi(n)) = 1$. Let G be the subgroup of $\mathbb{Z}^*_{n^2}$ with Jacobi symbol 1. Let K be the unique subgroup of order $2ab$, and H be the unique subgroup of order cdn.

Let g' be an element in $\mathbb{Z}^*_n$ of order $abcd$. Then $g = -(g')^n(1+n)$ computed in $\mathbb{Z}^*_{n^2}$ is an element of order $2abcdn$, that is, it generates G. We also note that it is easy to compute discrete logarithms in H, using a combination of Pohlig-Hellman and the Paillier method. This gives us a natural subgroup discrete logarithm problem $\mathcal{SDL}_{(G,K,H,g)}$.

Since $|H| = cdn$, this subgroup membership problem gives us a cryptosystem with higher bandwidth (message length divided by ciphertext length) than the Paillier cryptosystem. What can we say about the security of the cryptosystem? By Theorem 4, it is sufficient to consider the subgroup membership problem.

Theorem 7. *The above subgroup membership problem is hard if the Decision Composite Residuosity problem and the higher residue problem are both hard.*

Proof. Let H' be the subgroup of order cd, and let H'' be the subgroup of order n. By Theorem 1, $\mathcal{SM}_{(G,K)}$ is hard if $\mathcal{SM}_{(G,KH')}$ and $\mathcal{SM}_{(KH',K)}$ both are hard.

The former is the Decision Composite Residuosity problem. It is easy to see that the map $x \mapsto x^n \bmod n^2$ takes the higher residue problem to $\mathcal{SM}_{(KH',K)}$. Reduction modulo n takes $\mathcal{SM}_{(KH',K)}$ to the higher residue problem. Therefore, $\mathcal{SM}_{(G,K)}$ is hard if and only if both the Decision Composite Residuosity problem and the higher residue problem are hard. $\square$

Alternatively, if we are wary of trusting either problem fully, we can secret-share our message between the two subgroups H' and H''. An adversary must then solve both the Decision Composite Residuosity problem and the higher residue problem to break the cryptosystem. This variant is no longer homomorphic, however.

Paillier [14] also proposed a variant of his cryptosystem using (essentially) a subgroup of $\mathbb{Z}_n^*$. Let $p = 2ac + 1$, $q = 2bd + 1$ and $n = pq$, p, q, a, b, c and d prime, such that $\gcd(ac, bd) = \gcd(ab, cd) = \gcd(n, \phi(n)) = 1$. In this case, $\mathbb{Z}_{n^2}^*$ has unique subgroups of order ab and cd. Let G be the unique subgroup of $\mathbb{Z}_{n^2}^*$ of order abn, K be the subgroup of order ab and H be the subgroup of order n.

Consider G' to be the subgroup of quadratic residues of $\mathbb{Z}_n^*$. Let K' be the unique subgroup of order ab and H' be the subgroup of order cd. This group structure is similar to the higher residue group structure discussed in the previous section. Given a generator g for K', we get a subgroup membership problem $\mathcal{SM}_{(G',K')}$.

We claim that if the subgroup variant of Paillier's cryptosystem is weaker than the normal variant, then there exists a distinguisher for $\mathcal{SM}_{(G',K')}$.

First we note that the Decision Composite Residuosity problem is just as hard if restricted to the quadratic residues (since we can easily produce quadratic non-residues for the modulus n^2). Let G'' be the quadratic residues of $\mathbb{Z}_{n^2}^*$, let K'' be the subgroup of order $abcd$.

So we suppose that there is a distinguisher A for $\mathcal{SM}_{(G,K)}$ that fails on $\mathcal{SM}_{(G'',K'')}$. From A we shall produce a distinguisher for $\mathcal{SM}_{(G',K')}$.

Our distinguisher takes as input G' and $z \in G'$. It then samples b uniformly at random from $\{0, 1\}$, and computes $z' = z^n(1 + n)^{br}$ in $\mathbb{Z}_{n^2}$, for some random element $r \in \{1, \dots, n-1\}$. It gives A the element z', and A outputs a bit b'. If $b' = b$, we conclude that $z \in K'$, otherwise that $z \notin K'$.

It is clear that our distinguisher will work, since if $z \in K'$, z' will be in G'' and A will decide correctly significantly more often than not. If $z \notin K'$, then z' will not be in G'' and A will decide correctly roughly as often as it decides incorrectly. This means that our distinguisher will identify K' correctly more often than by guessing, and it will essentially be guessing when asked to identify $G' \setminus K'$.

We have proved the following result.

Theorem 8. *The subgroup variant is semantically secure if Paillier's cryptosystem is semantically secure, and the subgroup membership problem $\mathcal{SM}_{(G',K')}$ is hard.*

This allows us to transfer the trust we have in the subgroup membership problem $\mathcal{SM}_{(G',K')}$ to the subgroup variant of Paillier's cryptosystem, and we can concentrate our analysis on problems that are perhaps easier to deal with.

6 Concluding Remarks

We have given an abstract construction for a homomorphic public key cryptosystem, that has as special cases several famous homomorphic public key cryptosystems, including ElGamal, Goldwasser-Micali and Paillier. We have also formulated several general properties of subgroup membership problems, and shown how these general results aids in the analysis of new constructions.

Many useful and interesting results in cryptography essentially depend on subgroup membership problems as discussed in this paper. It is clear that the uniform treatment we have developed here can be extended to many of these results. Such work would clarify the thinking behind many schemes, and perhaps allow interesting generalisations to be made.

References

1. Josh Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 129–128, 1994.
2. D. Boneh. The Decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, volume 1423 of *LNCS*, pages 48–63. Springer-Verlag, 1998.
3. Mike Burmester, Yvo Desmedt, Fred Piper, and Michael Walker. A General Zero-Knowledge Scheme (Extended Abstract). In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Proceedings of EUROCRYPT '89*, volume 434 og *LNCS*, pages 122–133. Springer-Verlag, 1990.
4. Dario Catalano, Rosario Gennaro, and Nich Howgrave-Graham. The bit security of Paillier's encryption scheme and its applications. In Birgit Pfitzmann, editor, *Proceedings of EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 229–243. Springer-Verlag, 2001.
5. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Proceedings of EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer-Verlag, 2002.

6. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Proceedings of Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 119–136. Springer-Verlag, 2001.
7. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.
8. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.
9. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, April 1984.
10. Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In *Proceedings of ANTS IV*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.
11. D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In Nyberg [12], pages 308–318.
12. Kaisa Nyberg, editor. *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *LNCS*. Springer-Verlag, 1998.
13. T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In Nyberg [12], pages 308–318.
14. P. Paillier. Public-key cryptosystems based on composite degree residue classes. In Jacques Stern, editor, *Proceedings of EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.

A A Naccache-Stern Variant

Having a generator for H is sometimes desirable, motivating the following. Instead of having c and d be relatively prime, as above, we can let $c = d$, so that $p = 2ac + 1$, $q = 2bc + 1$, and $n = 4abc^2 + 2c(a + b) + 1$. Note that $c|(n - 1)$, so we may as well assume that c is known.

The interesting thing about the subgroup of order c^2 is that it is non-cyclic. If an element x of H is made public, and $x \neq 1 \pmod{p}$ and $x \neq 1 \pmod{q}$, then no matter what group operations are performed on x, the result will never yield a factor of n. It therefore seems safe to make elements of order c public.

Every element of order c generates a subgroup. If we have two elements x and y that generate distinct subgroups, we can use a Pollard ρ-type attack to generate a collision modulo p and not modulo q. This means that we should not publish elements from more than one subgroup.

In general, we will be interested in the modified group structure where K is the subgroup of order ab, H is a cyclic subgroup of order c generated by an element g, and $G = KH$. Note that we can sample elements from K by sampling elements uniformly from $\mathbb{Z}_n^*$ and raising them to the $2c$th power.

If we can recover the product ab, then we can factor n given any element m with Jacobi symbol -1 modulo n, because m^{abc} is congruent to 1 modulo one of the prime factors, and -1 modulo the other. Therefore, $\gcd(m^{abc} - 1, n)$ gives the factorisation of n.

First of all, $n - 1 \equiv 0 \pmod{c}$, so the attack of Naccache and Stern described above does not work. We get that

$$\frac{n-1}{4c^2} = ab + \frac{a+b}{2c}.$$

This means that ab is relatively close to $(n-1)/(4c^2)$. Note that $ac \approx bc \approx \sqrt{n}$. We can use a Baby-step Giant-step type algorithm to recover ab in

$$O(\sqrt{(a+b)/c}) = O(\sqrt{\sqrt{n}/c^2}) = O(n^{1/4}/c)$$

steps. This is not bad compared with (1). If c is kept sufficiently small, factoring n seems to be the best attack on the subgroup membership problem $\mathcal{SM}_{(G,K)}$.

Author Index