Lecture Notes in Computer Science 4058

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Lynn Margaret Batten
Reihaneh Safavi-Naini (Eds.)

Information Security and Privacy

11th Australasian Conference, ACISP 2006
Melbourne, Australia, July 3-5, 2006
Proceedings

Springer

Volume Editors

Lynn Margaret Batten
Deakin University
221 Burwood Highway, Burwood 3125, Victoria, Australia
E-mail: lmbatten@deakin.edu.au

Reihaneh Safavi-Naini
University of Wollongong
Centre for Information Security
Wollongong, NSW 2519, Australia
E-mail: rei@uow.edu.au

Library of Congress Control Number: 2006927435

CR Subject Classification (1998): E.3, K.6.5, D.4.6, C.2, E.4, F.2.1, K.4.1

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-35458-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-35458-1 Springer Berlin Heidelberg New York

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

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11780656 06/3142 5 4 3 2 1 0

Preface

The 11th Australasian Conference on Information Security and Privacy (ACISP 2006) was held in Melbourne, 3–5 July, 2006. The conference was sponsored by Deakin University, the Research Network for a Secure Australia, and was organized in cooperation with the University of Wollongong. The conference brought together researchers, practitioners and a wide range of other users from academia, industries and government organizations.

The program included 35 papers covering important aspects of information security technologies. The papers were selected from 133 submissions through a two-stage anonymous review process. Each paper received at least three reviews by members of the Program Committee, and was then scrutinized by the whole committee during a two-week discussion. There were 19 papers eligible for the "best student paper" award. The award was given to Yang Cui from the University of Tokyo for the paper "Tag-KEM from Set Partial Domain One-Way Permutations."

In addition to the regular papers the program also included three invited talks. Bart Preneel gave an invited talk entitled "Electronic Identity Cards: Threats and Opportunities." Mike Burmester's talk was "Towards Provable Security for Ubiquitous Applications." The details of the third talk had not been finalized at the time of publication of these proceedings.

We wish to thank all the authors of submitted papers for providing the content for the conference; their high-quality submissions made the task of selecting a program very difficult. We are indebted to the diligence and enthusiasm of the Program Committee members in ensuring selection of the most deserving papers and to the external reviewers who helped in the refereeing process. We wish to thank our sponsors, Research Network for a Secure Australia, for their support of the main speakers and students as well as Springer for their continued support of ACISP. We further wish to thank Judy Chow, the conference secretary, for her many organizational skills and patience with the registration process, and our Technical Chair, Jeffrey Horton, for his continuous effort and meticulous attention to every detail, which made the task of the Program Co-chairs so much easier.

Without the help of all the above this conference would not have been a possibility.

July 2006

Lynn Batten
Reihaneh Safavi-Naini

ACISP 2006

July 3–5, 2006, Melbourne, Australia

General Chair
Lynn Batten, Deakin University, Australia

Program Co-chairs
Lynn Batten, Deakin University, Australia
Reihaneh Safavi-Naini, University of Wollongong, Australia

Technical Chair
Jeffrey Horton, University of Wollongong, Australia

Program Committee

Tuomas Aura	Microsoft Research, UK
Feng Bao	Institute for Infocomm Research, Singapore
Colin Boyd	QUT, Australia
Liqun Chen	Hewlett-Packard Laboratories, UK
Kefei Chen	Shanghai Jiaotong University, China
Nicolas T. Courtois	Axalto Smart Cards, France
Robert Deng	Singapore Management University, Singapore
Marc Dacier	Eurecom Institute, France
Ed Dawson	QUT, Australia
Josep Domingo	University of Tarragona, Catalonia
Dieter Gollmann	Hamburg University of Technology, Germany
Juan Gonzalez Nieto	QUT, Australia
Goichiro Hanaoka	Nat. Inst. of Adv. Industrial Sci. and Tech., Japan
Markus Jakobsson	Indiana University, USA
Marc Joye	Gemplus & CIM-PACA, France
Tanja Lange	Technical University of Denmark, Denmark
Byoungcheon Lee	Joongbu University, Korea
Javier Lopez	University of Malaga, Spain
Subhamoy Maitra	Indian Statistical Institute, Kolkata, India
Catherine Meadows	Naval Research Lab, USA
Atsuko Miyaji	JAIST, Japan
Nasir Memon	New York Polytechnic, USA
SangJae Moon	Kyungpook National University, Korea
Keith Martin	Royal Holloway, University of London, UK
Peng Ning	North Carolina State University, USA
Kaisa Nyberg	Helsinki University of Technology and Nokia, Finland
Eiji Okamoto	Tsukuba University, Japan
Giuseppe Persiano	Università di Salerno, Italy
Josef Pieprzyk	Macquarie University, Australia
David Pointcheval	CNRS/ENS, Paris, France
Bimal Roy	Indian Statistical Institute, Kolkata, India
Palash Sarkar	Indian Statistical Institute, India

Jennifer Seberry	University of Wollongong, Australia
Juji Shikata	Yokohama National University, Japan
Nigel Smart	University of Bristol, UK
Douglas Stinson	University of Waterloo, Canada
Tim Strayer	BBN Technologies, USA
Clark Thomborson	University of Auckland, New Zealand
Serge Vaudenay	EPFL, Switzerland
Vijay Varadharajan	Macquarie University, Australia
Victor K. Wei	Chinese University of Hong Kong, Hong Kong

External Reviewers

Masayuki Abe
Joel Alwen
Nuttapong Attrapadung
Roberto M. Avanzi
Gildas Avoine
Thomas Baignères
Daniel J. Bernstein
Srimanta Bhattacharya
Olivier Billet
Mark Branagan
Emmanuel Bresson
Jaimee Brown
Billy Brumley
Debrup Chakraborty
Zhaohui Cheng
Andrew Clark
Christophe Clavier
Yvonne Cliff
Scott Contini
Yang Cui
Paolo D'Arco
Vanesa Daza
Ling Dong
Ratna Dutta
Stefan Dziembowski
Sarah Edwards
Mari Carmen Fernandez-Gago
Matthieu Finiasz
Eiichiro Fujisaki
Jun Furukawa
Clemente Galdi
Zheng Gong
Aline Gouget
Vanessa Gratzer
Jens Groth
JaeCheol Ha
Matt Henricksen
Jason Hinek
Xuan Hong
Zhenjie Huang
Sarath Indrakanti
Toshiyuki Isshiki
Tetsuya Izu
Christine Jones
Ari Juels
Lars Knudsen
Sandeep Kumar
Noboru Kunihiro
Kaoru Kurosawa
Eyal Kushilevitz
David Lapsley
Jens Ove Lauf
HoonJae Lee
Corrado Leita
Qiming Li
Ching Lin
Joseph Liu
Carl Livadas
Yu Long
Yi Lu
John Malone-Lee
Antoni Martínez-Ballesté
Sebastia Martin
Krystian Matusiewicz
Bill Millan
Hideyuki Miyake
Kunihiko Miyazaki
Jean Monnerat
Mridul Nandy
Stan Nurislov
Wakaha Ogata
Juan J. Ortega
Akira Otsuka
Vikram PAdman
Dan Page
Sylvain Pasini
Ahmed Patel
Kenny Paterson
Kun Peng
Pai Peng
Krzysztof Pietrzak
Jordi Castellà Roca
Rodrigo Roman
Kurt Rosenfeld
Chun Ruan
Naouel Ben Salem
Sumanta Sarkar
Francesc Sebé
Taha Sencar
Abdulattif Shikfa
SeongHan Shin
Leonie Simpson
Agustí Solanas
Masakazu Soshi
Ron Steinfeld
Gene Tsudik
Udaya Kiran Tupakula
Ivan Visconti
Martin Vuagnoux
Zhiguo Wan
Guilin Wang
Huaxiong Wang
Pan Wang
Ruizhong Wei
Mi Wen
Jian Weng
Christopher Wolf
Katsunari Yoshioka
Qinghua Zhang
Rui Zhang
Weiliang Zhao
Huafei Zhu

Table of Contents

Stream Ciphers

Algebraic Attacks on Clock-Controlled Stream Ciphers
Sultan Al-Hinai, Lynn Batten, Bernard Colbert, Kenneth Wong 1

Cache Based Power Analysis Attacks on AES
Jacques Fournier, Michael Tunstall 17

Distinguishing Attack on SOBER-128 with Linear Masking
Joo Yeon Cho, Josef Pieprzyk 29

Evaluating the Resistance of Stream Ciphers with Linear Feedback Against Fast Algebraic Attacks
An Braeken, Joseph Lano, Bart Preneel 40

Symmetric Key Ciphers

Ensuring Fast Implementations of Symmetric Ciphers on the Intel Pentium 4 and Beyond
Matt Henricksen, Ed Dawson 52

Improved Cryptanalysis of MAG
Leonie Simpson, Matt Henricksen 64

On Exact Algebraic [Non-]Immunity of S-Boxes Based on Power Functions
Nicolas T. Courtois, Blandine Debraize, Eric Garrido 76

Network Security

Augmented Certificate Revocation Lists
A. Lakshminarayanan, T.L. Lim 87

Online/Offline Signatures and Multisignatures for AODV and DSR Routing Security
Shidi Xu, Yi Mu, Willy Susilo 99

Towards an Invisible Honeypot Monitoring System
Nguyen Anh Quynh, Yoshiyasu Takefuji 111

Cryptographic Applications

Adaptively Secure Traitor Tracing Against Key Exposure and Its Application to Anywhere TV Service
Kazuto Ogawa, Goichiro Hanaoka, Hideki Imai 123

Fingercasting—Joint Fingerprinting and Decryption of Broadcast Messages
André Adelsbach, Ulrich Huber, Ahmad-Reza Sadeghi 136

More on Stand-Alone and Setup-Free Verifiably Committed Signatures
Huafei Zhu, Feng Bao 148

Secure Implementation

API Monitoring System for Defeating Worms and Exploits in MS-Windows System
Hung-Min Sun, Yue-Hsun Lin, Ming-Fung Wu 159

Hiding Circuit Topology from Unbounded Reverse Engineers
Yu Yu, Jussipekka Leiwo, Benjamin Premkumar 171

The Role of the Self-Defending Object Concept in Developing Distributed Security-Aware Applications
John W. Holford, William J. Caelli 183

Signatures

Efficient and Provably Secure Multi-receiver Identity-Based Signcryption
Shanshan Duan, Zhenfu Cao 195

Efficient Identity-Based Signatures Secure in the Standard Model
Kenneth G. Paterson, Jacob C.N. Schuldt 207

Event-Oriented k-Times Revocable-iff-Linked Group Signatures
Man Ho Au, Willy Susilo, Siu-Ming Yiu 223

Key Replacement Attack Against a Generic Construction of Certificateless Signature
Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, Xiaotie Deng 235

Theory

A Novel Range Test
Kun Peng, Colin Boyd, Ed Dawson, Eiji Okamoto 247

Efficient Primitives from Exponentiation in $\mathbb{Z}_p$
Shaoquan Jiang 259

PA in the Two-Key Setting and a Generic Conversion for Encryption with Anonymity
Ryotaro Hayashi, Keisuke Tanaka 271

Statistical Decoding Revisited
R. Overbeck 283

Invited Talk

Towards Provable Security for Ubiquitous Applications
Mike Burmester, Tri Van Le, Breno de Medeiros 295

Security Applications

Oblivious Scalar-Product Protocols
Huafei Zhu, Feng Bao 313

On Optimizing the k-Ward Micro-aggregation Technique for Secure Statistical Databases
Ebaa Fayyoumi, B. John Oommen 324

Provable Security

Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation Without Random Oracles
Eike Kiltz, David Galindo 336

Generic Transforms to Acquire CCA-Security for Identity Based Encryption: The Cases of FOPKC and REACT
Takashi Kitagawa, Peng Yang, Goichiro Hanaoka, Rui Zhang, Hajime Watanabe, Kanta Matsuura, Hideki Imai 348

Tag-KEM from Set Partial Domain One-Way Permutations
Masayuki Abe, Yang Cui, Hideki Imai, Kaoru Kurosawa 360

Protocols

An Extension to Bellare and Rogaway (1993) Model: Resetting Compromised Long-Term Keys
Colin Boyd, Kim-Kwang Raymond Choo, Anish Mathuria 371

Graphical Representation of Authorization Policies for Weighted Credentials
Isaac Agudo, Javier Lopez, Jose A. Montenegro 383

Secure Cross-Realm C2C-PAKE Protocol
Yin Yin, Li Bao 395

Hashing and Message Authentication

Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction
Praveen Gauravaram, William Millan, Ed Dawson, Kapali Viswanathan 407

Forgery and Key Recovery Attacks on PMAC and Mitchell's TMAC Variant
Changhoon Lee, Jongsung Kim, Jaechul Sung, Seokhie Hong, Sangjin Lee 421

Side Channel Attacks Against HMACs Based on Block-Cipher Based Hash Functions
Katsuyuki Okeya 432

Author Index 445

Algebraic Attacks on Clock-Controlled Stream Ciphers

Sultan Al-Hinai[1], Lynn Batten[2,⋆], Bernard Colbert[2], and Kenneth Wong[1]

[1] Information Security Institute (ISI)
Queensland University of Technology (QUT), Australia
[2] Deakin University, Australia

Abstract. We present an algebraic attack approach to a family of irregularly clock-controlled bit-based linear feedback shift register systems. In the general set-up, we assume that the output bit of one shift register controls the clocking of other registers in the system and produces a family of equations relating the output bits to the internal state bits. We then apply this general theory to four specific stream ciphers: the (strengthened) stop-and-go generator, the alternating step generator, the self-decimated generator and the step1/step2 generator. In the case of the strengthened stop-and-go generator and of the self-decimated generator, we obtain the initial state of the registers in a significantly faster time than any other known attack. In the other two situations, we do better than or as well as all attacks but the correlation attack. In all cases, we demonstrate that the degree of a functional relationship between the registers can be bounded by two. Finally, we determine the effective key length of all four systems.

Keywords: clock control, stream cipher, linear feedback shift register, irregular clocking, algebraic attack.

1 Introduction

Algebraic attacks, in which the generation of equations assists in determining the initial state or the key-stream of a cipher, were first applied to block ciphers and public key cryptosystems by Courtois and Pieprzyk [8, 13]. Algebraic attacks have been effectively applied to linear feedback shift register (LFSR) based systems as demonstrated in [1, 2, 6, 9, 10, 11, 12]. Our interest in this paper is their application to a class of bit-based LFSRs, which has not yet been examined from this direction — the irregularly clocked LFSR systems. We show that algebraic attacks are effective against this class of stream ciphers and we provide improvements to currently known attacks. In particular, known attacks against the Beth-Piper [3] strengthened stop-and-go cipher depend on the generation of weight three polynomials which cannot be done efficiently. We present a fast attack that is independent of the weight of the polynomial used.

⋆ This author wishes to thank the Australian Research Council for support of this work with a Discovery grant.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 1–16, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Irregular clocking in LFSRs is used to enhance their complexity and consequent security. Zenner [27] has developed a general approach to attacking such ciphers by guessing at the clocking through a clock cycle and applies this approach to several ciphers including A5/1, the stop-and-go generator, the alternating step generator and the step1/step2 generator with varying levels of success. Molland [20] introduces a general approach for dealing with LFSR systems with two registers where one register controls the clocking of the other. This applies to the basic stop-and-go generator, LILI-128 and step1/step2 generator. To our knowledge, the only algebraic attack on such a cipher is that on LILI-128 in [12], but guessing the clock control is an integral part of the approach.

Clock-controlled ciphers assume the existence of an underlying clock that maintains a consistent set of basic time intervals against which a register and its output can be compared. A bit-based LFSR system can then be established in a number of ways. A register can be stepped in synchrony with the underlying clock; it may move more slowly than the underlying clock, taking more than one basic unit to shift the registers; but it can be assumed that it will never shift faster than the clock as otherwise we can adjust the basic clocking time to the step time of the register. Similarly, the output from a system of clock-controlled LFSRs can be synchronized with the clock time or can be slowed down or varied against the clock time. If a register shifts with the basic time interval, we refer to is as *regularly clocked*. If the output is delivered with the basic time interval, we refer to it as *regular output*.

We shall focus on four systems, which fall into the above types. The Beth-Piper strengthened stop-and-go generator uses two regularly clocked registers and one irregularly clocked register along with regular output. The alternating step generator has one regularly clocked register and two which are irregularly clocked. Again, the output is regular. In the self-decimated generator, the output is irregular while the sole register actually clocks regularly. The step1/step2 generator has two irregularly clocked LFSRs with irregular output.

In the next section, we introduce the notation standardized throughout the paper and present our generic approach to determining algebraic equations involving the initial state bits from bit-based clock-controlled ciphers in which a linear combination of bits from regularly clocked registers determines the clocking of the others. Our aim is to recover the initial state bits. We ignore key initialization schemes altogether and compute effective key length assuming that the key and initial state of the registers are one and the same. We therefore use the phrase *effective key length*, and make the assumption that for computational purposes eighty bit registers are safe from a brute force attack. In subsequent sections, we use the equations to find the initial states of the four ciphers mentioned above. In the case of the strengthened stop-and-go generator and of the self-decimated generator, we obtain the initial state of the registers in a significantly faster time than any other known attack. In the other two situations, we do better than or as well as all attacks but the correlation attack. In all cases, we demonstrate relationships between the registers indicating that a low degree multiple of the polynomials corresponding to irregularly clocked registers can

be bounded by two. Finally, we determine the effective key length of all four systems and present our computational results.

2 The General Set-Up

We consistently label registers A, B and C with lengths l, m and n. The i^{th} bit of register A at time t is denoted by A_i^t and so the output bit is A_l^t. Similarly for registers B and C. We use z^t to denote the output from the entire system at time t. M denotes the number of monomials occurring in a given system of equations. If this system is linear, the complexity of solving the system is in general about M^3; if the system is sparse, this reduces to M^2 [25]. If the system is quadratic, the complexity of using the linearisation methods in [12] is about $\binom{M}{2}^3$. In this paper, we make use of both linear algebra and Gröbner bases methods of solving equations. All computations are performed using the F_4 algorithm in Magma 2.11 [19] on the SGI Origin 3000 using CPU at 600 MHz.

In setting up a general approach to acquiring an equation from a clock-controlled stream cipher of our type, we need to consider three things: first of all, which LFSR controls the clocking (we always use the letter A for this register), secondly, which bits of the controlling register are used to determine the clocking, and thirdly, the effect on the shift of the controlled register. Suppose that the i^{th} bit of A controls the clocking of B in such a way that if it is 0, B does not clock and if it is 1, B clocks j times. In this case, we can express the change to the k^{th} position of B as follows:

$$B_k^t = B_k^{t-1}(A_i^{t-1} \oplus 1) \oplus B_{k-j}^{t-1} A_i^{t-1}. \tag{1}$$

Of course, this applies if $(k - j) > 0$ as otherwise, we need to accommodate the feedback polynomial into the equation, which is easily done. Modifications can also be made to take into consideration the use of several bits of A being used to determine the clocking of B and in cases where more than one register is used in determining the clocking of other registers. As we shall see in the basic stop-and-go generator, a system as simple as the above is inherently weak because if the sum of two consecutive output bits is 1, we already get information about bits in register A. In the more general situation of a polynomial P in the bits of A controlling the clocking of the register B, equation (1) becomes

$$B_k^t = B_k^{t-1}(P \oplus 1) \oplus B_{k-j}^{t-1} P. \tag{2}$$

We state and prove the following theorem involving several regularly clocked registers A_i.

Theorem. *Consider a bit-based LFSR system with k regularly clocked LFSRs A_i of length l_i respectively, $1 \leq i \leq k$, in which a linear polynomial L involving bits of the A_i determines the clocking of a register B of length m as described in (2). Suppose the output z^t at time t is the binary sum of the outputs of all registers. Then the initial state of all A_i can be recovered from a system of quadratic equations, and can subsequently be used to recover the output of B.*

Proof. We have $z^t = B_m^t \oplus \sum_i A_{l_i}^t$, and using (2), can write

$$z^{t+1} = B_m^t(L \oplus 1) \oplus B_{m-j}^t L \oplus \sum_i A_{l_i}^{t+1}. \tag{3}$$

$$\text{Therefore,} \qquad z^t \oplus z^{t+1} = L(B_m^t \oplus B_{m-j}^t) \oplus \sum_i (A_{l_i}^t \oplus A_{l_i}^{t+1}). \tag{4}$$

Multiplying by $L \oplus 1$, this results in an equation of degree at most two involving the bits of the A_i:

$$(L \oplus 1)(z^t \oplus z^{t+1}) = (L \oplus 1) \sum_i (A_{l_i}^t \oplus A_{l_i}^{t+1}). \tag{5}$$

This quadratic system can be solved for all bits of the registers A_i by running off sufficiently many output bits from the system. The output of B can then be calculated from

$$B_m^t = z^t \oplus \sum_i A_{l_i}^t. \qquad \square$$

The above theorem indicates that using more than one regularly clocked register in a linear way to produce the output of the system adds no additional security, as one sufficiently long such register will suffice. This confirms, as a special case, the result of [23]. Although the assumptions of the Theorem apply only to the stop-and-go generator in our list of target ciphers, the method used in the proof applies to a general class of clock-controlled LFSR based systems, those in which a linear function of register bits controls the clocking of several registers. The alternating step and step1/step2 ciphers fall into this category. As we shall see in section 4, the method also works on a system with only one register - the self-decimated generator.

The generation of the equations (sufficiently many to be able to derive a solution from them for the unknowns) is independent of the register values, and so is assumed to be a precomputation procedure. In all cases below, we use a maximum of approximately 2 GB of memory for equation generation. While A produces linear equations from its feedback polynomial, B is producing higher degree equations and so its output is always used in the output of the entire system, which is therefore also highly non-linear. Our aim is therefore to reduce the high degree of the output equations. In each of the four systems discussed below, we take combinations of consecutive output bits in order to obtain reduced degree equations. We compare our attack against each system with other best attacks, based on the keystream requirements, the attack complexity, and the precomputation complexity. We also determine the effective key length for securing the system against the attacks.

3 Beth-Piper Stop-and-Go Generator — Strengthened Version

The basic Beth-Piper stop-and-go generator [3], has two LFSRs A and B of lengths l and m respectively. A is regularly clocked and its output controls the clocking of B. Register B is clocked if and only if the output of A is 1. Rueppel [22] and Kanso [18] point out that the generator has statistical weaknesses. They observed that when the binary derivative output $z^t \oplus z^{t+1}$ is equal to 1, then A_l^t has to be 1.

Because of the weakness of the basic stop-and-go, we will move directly to an analysis of the strengthened stop-and-go generator. However, we wish first to describe in detail our general technique for generating equations on the basic system.

From equation (1) , we produce equations as follows for two consecutive output bits of the system.

$$z^t = B_m^t = B_m^{t-1} A_l^{t-1} \oplus B_m^{t-1} \oplus B_{m-1}^{t-1} A_l^{t-1}, \tag{6}$$

$$z^{t+1} = B_m^t A_l^t \oplus B_m^t \oplus B_{m-1}^t A_l^t. \tag{7}$$

Adding (6) and (7) yields

$$z^t \oplus z^{t+1} = B_m^{t-1} A_l^{t-1} \oplus B_m^{t-1} \oplus B_{m-1}^{t-1} A_l^{t-1} \oplus B_m^t A_l^t \oplus B_m^t \oplus B_{m-1}^t A_l^t. \tag{8}$$

Substituting for B_m^t in the right hand side of (8) we obtain

$$z^t \oplus z^{t+1} = A_l^t (B_m^t \oplus B_{m-1}^t). \tag{9}$$

From equation (9), we observe easily, as did Kanso and Rueppel, that when the left side is 1, then A_l^t is 1. However, we also obtain the fact that $B_m^t \oplus B_{m-1}^t$ equals 1 in this case, giving us new information about the bit B_{m-1}^t in the irregularly clocked register which differs by 1 from the output bit z^t. Note also that in (9) the right-hand side is a product of a linear term with a high degree term and this is reduced to the constant on the left-hand side. Consequently, running off about l keystream bits gives enough equations to solve the sparse linear system in A with a complexity of l^2. Once the initial state of A is determined, we can then use the output from the system to determine the initial state of B directly.

Molland [20] points out that his method applies to the basic stop-and-go generator, but with a complexity of order 2^l. His method does not apply to the strengthened stop-and-go generator which we now consider.

3.1 The Strengthened Version

The strengthened version of the Beth-Piper stop-and-go generator [3] as shown in Figure 1, employs three LFSRs; in addition to A and B as in the basic stop-and-go, it uses a third regularly clocked register C of length n. The output bit at time t is formed by combining, using XOR, the sequences from B and C. A

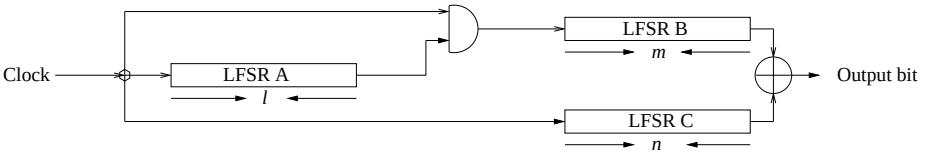


Fig. 1. The Beth-Piper Stop-and-Go Generator - Strengthened Version

thorough analysis of the cipher has been conducted in [18], confirming that the cipher generates sequences with good statistical properties.

The one significant attack on the strengthened version of the stop-and-go generator is due to [26], which uses an improved linear syndrome algorithm. The approach uses trinomial multiples of the feedback polynomial of each LFSR to generate syndromes which are equations in the output bits. It is assumed that the feedback polynomials are primitive trinomials. Under this assumption, the attack is very fast, solving the problem in $896 \max(l, m)$ computations based on $37 \max(l, m)$ keystream bits. When a feedback polynomial is not trinomial, the linear syndrome method relies on being able to replace it by its trinomial multiple of the least degree. The best-known method for determining such multiples is due to Coppersmith [7,24] and only works when the register length is a power of two as the approach relies on determining discrete logarithms in fields of characteristic 2. We note that our algebraic attack method is applicable regardless of the weight of the feedback polynomial.

3.2 Algebraic Attack on the Strengthened Version

We apply the same method as that used for the basic stop-and-go generator. The output of the irregularly clocked register is given by

$$B_m^t = B_m^{t-1}(A_l^{t-1} \oplus 1) \oplus B_{m-1}^{t-1} A_l^{t-1}. \tag{10}$$

The output is then

$$z^t = B_m^t \oplus C_n^t. \tag{11}$$

The maximum degree an equation can have for this system is $l+1$. The number of monomials is $M = m(2^l - 1) + n + 1$ and the complexity of solving this system of equations using the standard linearization method is then M^3 which is greater than exhaustive keysearch.

3.3 Reducing the Degree of the Equations

In this section we reduce the degree of the equations generated by the irregularly clocked generator. The method of degree reduction is basic to the algebraic attack method. From (9),

$$z^{t+1} = B_m^{t+1} \oplus C_n^{t+1} = B_m^t A_l^t \oplus B_m^t \oplus B_{m-1}^t A_l^t \oplus C_{n-1}^t. \tag{12}$$

Combining this with (11) yields

$$z^t \oplus z^{t+1} = (B_m^t \oplus B_{m-1}^t)A_l^t \oplus C_n^{t+1} \oplus C_n^t. \tag{13}$$

We eliminate the high degree terms in this equation by multiplying both sides by $A_l^t \oplus 1$.

$$(A_l^t \oplus 1)(z^t \oplus z^{t+1}) = A_l^t C_n^{t+1} \oplus A_l^t C_n^t \oplus C_n^t \oplus C_n^{t+1}. \tag{14}$$

Equation (14) determines a system of quadratic equations in the bits of regularly clocked registers A and C. In determining the effective key length, we use 2^{80} as a rule of thumb value for resistance to brute force attacks. Because the system of equations generated may well be sparse and therefore computable in square rather than cube time, we recommend register lengths of about 2^{15} as a minimum. We applied algebraic attacks using Gröbner bases methods to solve the system of equations obtained from the cipher.

Table 1. Best Known Attacks on Beth-Piper Stop-and-Go

Attack	**Minimum keystream required MK**	**MK** $l = m =$ $n = 64$	**pre computation complexity**	**Solution complexity**	**Total attack complexity TC**	**TC** $l = m =$ $n = 64$	**TC** $l = m =$ $n = 128$
Linear syndrome attack [26]	$37 \max(l, n)^\star$	2^{12}	$2^{n^{2/3}+2n^{2(n^{1/3}-1)/3}}$ to ** $2^{n^{1/3}+2n^{(4n^{2/3}-1)/3}}$	$(l+n)^3$ to $(l+n)^2$	$2^{n^{2/3}+2n^{2(n^{1/3}-1)/3}}$ $+(l+n)^3$	2^{30} to 2^{132}	2^{47} to 2^{241}
Our attack Gröbner	$l+n$	2^7	$O(l^2)$	Gr	Gr	Gr	Gr

Table 1 presents both for ours and other attacks the minimum keystream required, precomputation complexity, where applicable, solution complexity, and the total attack complexity which is the combination of precomputation and solution complexities; where there is essentially no precomputation (reusable) work, we leave this spot blank. We also give values of the keystream and the total complexity for register sizes 64 and/or 128. The entry Gr indicates that complexity of the work using a Gröbner bases method is generally unknown. To obtain a measure of this complexity we obtained empirical results from experimental attacks (see Section 3.4).

It should be noticed that it is not useful to solve the system obtained from equation (14) with Gaussian elimination by linearisation of the system. This is because when we multiply the equations by $A_l^t \oplus 1$, we introduce a new set of monomials that need to be linearised into more variables, while the rank of the system stays constant. The system then becomes underdetermined and hence cannot be solved for a unique solution. We solve the system by Gröbner bases methods instead.

* This only applies for LFSRs with trinomial primitive polynomials, in which case the attack takes time $896 \max(l, m)$. Trinomial Polynomials should not be used.

** The lower precomplexity corresponds to higher solution complexity and higher precomplexity to lower solution complexity in this table. The total attack complexity averages these out. The total attack complexity is then derived from the best case.

3.4 Implementation of the Gröbner Bases Methods on the Beth-Piper Stop-and-Go Generator

From equation (14) we obtain quadratic equations in terms of initial state bits of registers A and C, with $l+n$ variables. When we used the F_4 algorithm of the Gröbner bases methods, we obtained a unique solution to the initial states of A and C. This means that the minimum keystream required for the attack is $l+n$. In practice, we usually need significantly more than $l+n$ equations for finding a unique solution. It can be seen from Table 2 that the more keystream we have, the more efficient it is to find a solution of the system of equations we generate. Performing multiplication on the algebraic equations introduces dependencies

Table 2. Attack Times for Beth-Piper Stop-and-Go Generator

l, n	**Number of variables**	**Keystream used**	**Time to generate equations**	**Time to find solution**
16 bit	32	128 bit	0.8 s	32 s
16 bit	32	256 bit	1.6 s	0.75 s
32 bit	64	1024 bit	40 s	60 s
32 bit	64	1536 bit	60 s	10 s
64 bit	128	4096 bit	513 s	3889 s
64 bit	128	5120 bit	649 s	618 s

and so the attack requires about $l^2/2$ bits of keystream to find unique solutions. These results indicate that this method is significantly better than the technique proposed in [26].

4 Self-decimated Generator

The (d, k) self-decimated generator, proposed by Rueppel [21] and shown in Figure 2, consists of a single LFSR A which controls its own clocking on the principle that if the output is 0, then A clocks d times before producing new output, and if the output is 1, A clocks k times before doing so.

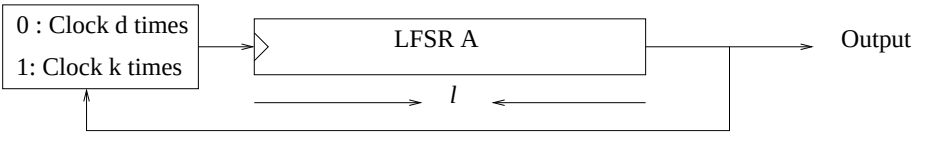


Fig. 2. The Self-Decimated Generator

Rueppel has proved [21] that a (d, k) self-decimated generator is theoretically equivalent to a $(1, k)$ self-decimated generator in combination with a generator regularly clocked in intervals of length d. In [4], it was pointed out that if d and

k are at most 3, the generator might be subject to a correlation attack. Though we know of no such attack, we present here an attack on a $(1, 4)$ self-decimated generator. However, the situation can be generalized fully to the $(1, k)$ case. Equation (1) gives for $5 \leq i \leq l$ and $2 \leq j \leq 4$

$$A_i^t = A_{i-1}^{t-1}\left(A_l^{t-1} \oplus 1\right) \oplus A_{i-4}^{t-1}A_l^{t-1} \tag{15}$$

$$A_j^t = A_{j-1}^{t-1}\left(A_l^{t-1} \oplus 1\right) \oplus F_{l-(4-j)}^{t-1}A_l^{t-1}$$

$$A_1^t = F_l^{t-1}\left(A_l^{t-1} \oplus 1\right) \oplus F_{l-3}^{t-1}A_l^{t-1}$$

where F_j^{t-1} applies the feedback polynomial from the $j'th$ position at time $t-1$, and incorporates knowledge of previous bits. The generator will produce equations of maximum degree $l+1$ where l is the length of the underlying LFSR, and the number of monomials that the system of equations can have is therefore equal to $M = 2^l - 1$. Thus the maximum attack complexity is M^3, using the linearisation methods of [12]. As in the method of the theorem, using (1), the output of the generator at time $t+1$ is

$$z^{t+1} = A_l^{t+1} = A_{l-1}^t(A_l^t + 1) \oplus A_{l-4}^t A_l^t. \tag{16}$$

This can be rewritten as

$$z^{t+1} = A_{l-1}^t z^t \oplus A_{l-4}^t z^t \oplus A_{l-1}^t. \tag{17}$$

Thus the knowledge of two consecutive keystream bits gives us a linear equation in the bits of A. Our complexity, as described in Table 3, shows that $O(l)$ keystream bits is sufficient to get a system of linear equations solvable for the initial state. The effective key length is 2^{30}. Table 3 presents for our attack the minimum keystream required, precomputation complexity, where applicable, solution complexity, and the total attack complexity which is the combination of precomputation and solution complexities. We also give values of the keystream and the total complexity for register sizes 64 and/or 128.

Table 3. Best Known Attacks on the Self-Decimated Generator

Attack	**Minimum keystream required MK**	**MK** $l = 128$	**pre computation complexity**	**Solution complexity**	**Total attack complexity TC**	**TC** $l = 128$
Our attack	$O(l)$	2^7	$O(1)$	$O(l^3)$	$O(l^3)$	2^{21}

Rueppel [22] notes that since this generator outputs directly bits of A, this reveals information about A and so could be weak. He suggests the possibility of out-putting a linear combination of bits of A. We consider, as in equation (2), the situation where a polynomial P in bits A_i of A is added to the output:

$$z^{t+1} = P(A_i^t) \oplus A_{l-1}^t z^t \oplus A_{l-4}^t z^t \oplus A_{l-1}^t.$$

Table 4. Algebraic Attack Times for Self-Decimated Generator

l	Number of variables	Keystream used	Time to generate equations	Time to find solution
128 bit	128	160 bit	6 s	0.07 s
256 bit	256	384 bit	71 s	0.6 s
512 bit	512	640 bit	182 s	3 s

If P is linear, the system is subject to attack with the same complexity as the original system. As the degree of P increases, so does the complexity.

We have implemented the attack on the self-decimated generator for different register lengths. From equation (17) we obtain linear equations in terms of the initial state bits of register A, with l variables in constant time. Given l linearly independent equations, Gröbner bases methods are equivalent to Gaussian elimination. In practice, we usually need more than l equations to yield a linear system of full rank. Equations can be generated in linear time and the system can be solved in $O(l^3)$. It can be seen from Table 4 that the self-decimated generator can be broken in polynomial time, and for standard register lengths, the initial state of the generator can be found efficiently using keystream of order l.

5 Step1/Step2 Generator

Gollmann and Chambers proposed the step1/step2 generator in [16]. As shown in Figure 3, the generator consists of two LFSRs, A and B, which are of the same length. A controls the shifting of B in the following way. If the output of A is 0, then B is clocked once and if the output of A is 1, B is clocked twice before producing the keystream bits. Note that if A outputs 1, it must wait for B to clock twice and hence does not regularly clock. An alternate approach to considering this system such that A is regularly clocked is to have A move twice every clock beat and consequently B will move either once or twice every clock beat. The output of the system is the output of B. There have been three substantial attacks on this generator [28, 27, 20].

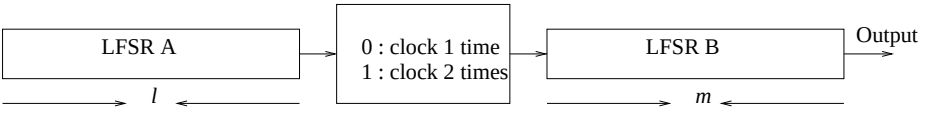


Fig. 3. The Step1/Step2 Generator

5.1 Algebraic Analysis of the Step1/Step2 Generator

In this section, we algebraically analyse the step1/step2 generator. Using the general equation of Section 2, each stage i of B for $2 < i < m$ is replaced by the following expression.

Table 5. Best Known Attacks on Step1/Step2 Generator

Attack	**Minimum required MK**	**MK** $l = m = 64$	**pre computation complexity**	**Solution complexity**	**Total attack TC**	**TC** $l = m = 64$
Embedding corr'n attack [28]	$5m$	2^8	$2^m m$	up to m^3	$> 2^m 5m$	2^{72}
Clock control guessing attack [27]	$l + m$	2^7	—	$O((l+m)^3 2^{(l+m)/2})$	$O((l+m)^3 2^{(l+m)/2})$	2^{84}
Improved LCT attack [20]	$3l/2$	2^7	$> O(l^9)$	$O(2^l)$	$O(2^l)$	2^{64}
Our attack Gaussian	l	2^7	—	$2^l m^3$	$2^l m^3$	2^{82}
Our attack Gröbner	$l + m$	2^7	$O(2^l)$	Gr	Gr	Gr

$$B_i^t = B_{i-1}^{t-1}(A_l^{t-1} \oplus 1) \oplus B_{i-2}^{t-1} A_l^{t-1} \tag{18}$$

and B_2 is given by

$$B_2^t = B_1^{t-1}(A_l^{t-1} \oplus 1) \oplus F_m^{t-1} A_l^{t-1}. \tag{19}$$

The right-most position is given by

$$B_1^t = F_m^{t-1}(A_l^{t-1} \oplus 1) \oplus F_{m-1}^{t-1} A_l^{t-1} \tag{20}$$

where F_j^{t-1} applies the feedback polynomial from the $j'th$ position at time $t-1$. Since A controls the clocking of B, the maximum degree an equation can have is $l+1$ and $M = m(2^l - 1) + 1$. The output is given by

$$z^t = B_m^t = B_{m-1}^{t-1} A_l^{t-1} \oplus B_{m-1}^{t-1} \oplus B_{m-2}^{t-1} A_l^{t-1}. \tag{21}$$

We add two consecutive output bits $z^t = B_m^t$ and $z^{t+1} = B_{m-1}^t(A_l^t \oplus 1) \oplus B_{m-2}^t A_l^t$ to get

$$z_t \oplus z_{t+1} = (B_{m-1}^t \oplus B_{m-2}^t) A_l^t \oplus B_{m-1}^t \oplus B_m^t. \tag{22}$$

Multiplying through by $A_l^t \oplus 1$ produces

$$(A_l^t \oplus 1)(z^t \oplus z^{t+1}) = (A_l^t \oplus 1)(B_{m-1}^t \oplus B_m^t). \tag{23}$$

Thus a linear multiple of the complex component in B reduces to a low degree function on the left. We can now guess register A bits and solve the resulting system in B with complexity $2^l m^3$, using the linearisation methods of [12]. Effective key length is 80. Table 5 presents both for ours and other attacks the minimum keystream required, precomputation complexity, where applicable, solution complexity, and the total attack complexity which is the combination of precomputation and solution complexities; where there is essentially no precomputation (reusable) work, we leave this spot blank. We also give values of the keystream and the total complexity for register sizes 64 and/ or 128.

Table 6. Algebraic Attack Times for Step1/Step2 Generator

l, m	Number of variables	Keystream used	Time to generate equations	Time to find solution
8 bit	16	16 bit	13 s	3 s
10 bit	20	20 bit	936 s	89 s
12 bit	24	24 bit	153699 s	5958 s

Table 5 gives separate rows for complexities derived using linear algebra methods and respectively Gröbner bases methods. In Table 5, the precomputation complexity shown in the last line is derived from polynomial multiplication. The entry Gr indicates that complexity of the work using a Gröbner bases method is generally unknown. We provide empirical data derived from using the Gröbner bases method as shown in Table 6. Since multivariate polynomial multiplication is exponential in the number of variables, we restricted bit sizes to values we were able to run in under two days as can be seen in Tables 6 and 8.

5.2 Implementation of the Gröbner Bases Methods on the Step1-Step2 Generator

If we do not guess any bits of A or B, we would generate a system of equations of degree $l+1$ with $l+m$ variables. Using Gröbner bases methods we can obtain a unique solution to the initial states of A and B. This means that the minimum keystream required for the attack is $l+m$. In Table 6 we generate the equations without guesses at bits of A; nevertheless we can still recover the initial states with the minimum required keystream. Although the computations of Section 5.1 indicate that a larger size for A increases the complexity faster, we have implemented attacks with $l = m$.

6 Alternating Step Generator

The alternating step generator was introduced in [17] and is shown in Figure 4. It employs three LFSRs A, B and C of lengths l, m and n respectively. A is regularly clocked and controls the clocking of B and C. If the output of A at time t is 1, then B is clocked; otherwise, C is clocked. The output of the system is the sum of the outputs of registers B and C. Several attacks have been applied to the alternating step generator [14, 15, 27] with complexity indicated in Table 7.

6.1 Algebraic Attack on the Alternating Step Generator

Using the general equation of Section 2,

$$B_i^t = B_i^{t-1}(A_l^{t-1} \oplus 1) \oplus B_{i-1}^{t-1} A_l^{t-1} \tag{24}$$

$$C_i^t = C_i^{t-1} A_l^{t-1} \oplus C_{i-1}^{t-1}(A_l^{t-1} \oplus 1). \tag{25}$$

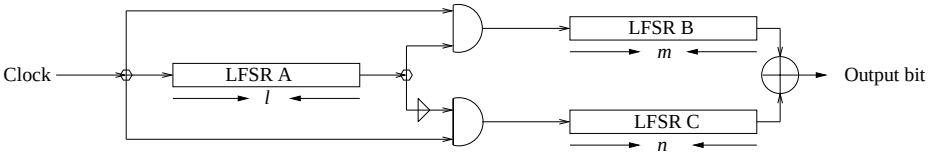


Fig. 4. The Alternating Step Generator

Table 7. Best Known Attacks on the Alternating Step Generator

Attack	**Minimum keystream required MK**	**MK** $l = m = n = 64$	**pre computation complexity**	**Solution complexity**	**Total attack complexity TC**	**TC** $l = m = n = 64$
Edit distance corr'n [14]	$O(m+n)$	2^9	—	$O(2^{m+n}(m+n))$	$O(2^{m+n}(m+n))$	2^{135}
Clock control guessing attack [27]	$l+m+n$	2^8	—	$O((l+m+n)^3 2^{(l+m+n)/2})$	$O((l+m+n)^3 2^{(l+m+n)/2})$	2^{119}
Improved edit distance corr'n [15]	$O(\max(m,n))$	2^{11}	—	$O(2^{\max(m,n)}\max(m,n))$	$O(2^{\max(m,n)}\max(m,n))$	2^{70}
Our attack Gaussian	$\max(m,n)$	2^7	—	$O(2^l(n^3+m^3))$	$O(2^l(n^3+m^3))$	2^{83}
Our attack Gröbner	$l+\max(m,n)$	2^8	$O(2^l)$	Gr	Gr	Gr

The keystream bit at time t is then $z^t = B_m^t \oplus C_n^t$. The maximum degree an equation can have is $l+1$ and the number of monomials is $M = (n+m)(2^l - 1) + n + 1$.

In applying our algebraic attack, we again use the method of the theorem of Section 2 to obtain

$$z^t \oplus z^{t+1} = B_m^{t+1} \oplus B_m^t \oplus C_n^{t+1} \oplus C_n^t \tag{26}$$

which can be rewritten as

$$z^t \oplus z^{t+1} = (B_m^t \oplus B_{m-1}^t)A_l^t \oplus (C_n^t \oplus C_{n-1}^t)(A_l^t \oplus 1). \tag{27}$$

Multiplying both sides of (27) by $A_l^t \oplus 1$ produces

$$(A_l^t \oplus 1)(z^t \oplus z^{t+1}) = (A_l^t \oplus 1)(C_n^t \oplus C_{n-1}^t). \tag{28}$$

Thus, a linear multiple of the complex component in C reduces to a low degree function on the left. We can guess register A bits and solve the resulting system in C with complexity $2^l n^3$. Then equation (26) can be used to solve for the bits of B with additional complexity m^3. The total complexity is $2^l(n^3+m^3)$. Alternatively, we can guess register C, recover the bits of A and then B with total complexity $2^n(l^3+m^3)$. Guessing the shortest register is clearly the best option. The effective key length is 2^6.

Table 7 presents both for ours and other attacks the minimum keystream required, precomputation complexity, where applicable, solution complexity, and

the total attack complexity which is the combination of precomputation and solution complexities; where there is essentially no precomputation (reusable) work, we leave this spot blank. We also give values of the keystream and the total complexity for register sizes 64 and/or 128. Table 7 gives separate rows for complexities derived using linear algebra methods and respectively Gröbner bases methods. In Table 7, the precomputation complexity shown in the last line is derived from polynomial multiplication. The entry Gr indicates that complexity of the work using a Gröbner bases method is generally unknown. We provide empirical data derived from using the Gröbner bases method as shown in Table 8.

6.2 Implementation of the Gröbner Bases Methods on the Alternating Step Generator

If we do not guess any bits of A, B or C, we would generate a system of equations of degree $l + 1$ with $l + m + n$ variables. Using Gröbner bases methods we can obtain a unique solution to the initial states of A and B. This means that the minimum keystream required for the attack is $l + m + n$. It is shown in Table 8 that we can recover the initial states with the minimum required keystream.

Table 8. Algebraic Attack Times on Alternating Step Generator

l, m	**Number of variables**	**Keystream used**	**Time to generate equations**	**Time to find solution**
8 bit	16	24 bit	27 s	7 s
10 bit	20	30 bit	2706 s	1830 s

7 Conclusions and Suggestions for Future Research

We have presented an algebraic method of attacking a general type of clock-controlled bit-based stream cipher along with a theorem bounding by two the degree of the derived equations from the cipher. The method of the theorem can be used for a broader range of ciphers in determining their levels of security, and we have demonstrated its effectiveness on the strengthened stop-and-go, self-decimated, step1/step2 and alternating step generators. In comparing the efficiency of our attacks with other attacks, we observed that when the feedback polynomial is not a trinomial, the known attacks generate trinomial polynomials which is a highly complex procedure. Our results, however, do not depend on constructions of polynomials. For the Beth-Piper stop-and-go generator strengthened version and for the self-decimated generator, we obtain significantly better attacks than any other known attack. For the other two ciphers, our attack complexity is not far from the best known attacks. For each of the generators studied, we have presented the effective register length of the system providing useful information for implementations.

This paper is the first presenting a general algebraic attack approach to clock-controlled bit-based stream ciphers in which the output of a single LFSR (or a

linear combination of its bits) determines the clocking of other LFSRs in the system. Our results indicate that long register lengths are needed to protect against algebraic attacks. In future work, we will develop the technique for use against other classes of clock-controlled ciphers including the cascade ciphers [5].

Acknowledgments. The authors wish to express their appreciation to Ed Dawson, Matt Henricksen, William Millan and Leonie Simpson for helpful discussions on this work, to the High Performace Computing and Research Support at Queensland Univeristy of Technology for providing us hardware and assistance with our computational experiements, and also to the Computational Algebra Group at the University of Sydney for a complimentary copy of Magma for use on our hardware.

References

1. F. Armknecht. Improving fast algebraic attacks. *FSE*, pages 65–82, 2004.
2. F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. *Crypto*, pages 162–175, 2003.
3. T. Beth and F. C. Piper. The stop-and-go generator. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology: Proceedings of Eurocrypt 84*, volume 209 of *Lecture Notes in Computer Science*, pages 88–92. Springer-Verlag, 1985.
4. W. G. Chambers and D. Gollmann. Embedding attacks on step[1..D] clock controlled generators. *Electronics Letters*, 36, pages 1771–1773, 2000.
5. W. G. Chambers and D. Gollmann. Lock-in Effect in Cascades of Clock-Controlled Shift-Registers. In Christoph G. Günther,editor,*Advances in Cryptology: Proceedings of Eurocrypt 88*,volume 330 of *Lecture Notes in Computer Science*, pages 331–344. Springer-Verlag, 1988.
6. J. Y. Cho and J. Pieprzyk. Algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering. *FSE*, pages 49–64, 2004.
7. D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30(4), pages 587–593, 1984.
8. N. Courtois. The security of hidden field equations (HFE). *CT-RSA*, pages 266–281, 2001.
9. N. Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt. *ICISC*, pages 182–199, 2002.
10. N. Courtois. Algebraic attacks on combiners with memory and several outputs. Cryptology ePrint Archive, Report 2003/125, 2003.
11. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. *Crypto*, pages 176–194, 2003.
12. N. Courtois and W. Meier. Algebraic attacks on stream cipher with linear feedback. In *Advances in Cryptology - Eurocrypt - LNCS 2656, Springer-Verlag*, pages 346–359, 2003.
13. N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. *Asiacrypt*, pages 267–287, 2002.
14. J. Dj. Golić and R. Menicocci. Edit distance correlation attack on the alternating step generator. In Burton S. Kaliski Jr., editor, *Advances in Cryptology–Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 499–512. Springer-Verlag, 1997.

15. J. Dj. Golić and R. Menicocci. Correlation analysis of the alternating step generator. *Des. Codes Cryptography*, 31(1), pages 51–74, 2004.
16. D. Gollmann and W. G. Chambers. Clock-controlled shift registers: a review. *IEEE Journal on Selected Areas in Communications*, 7 (1989), pages 525–533, 1989.
17. C. G. Günther. alternating step generators controlled by de Bruijn sequences. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology–Eurocrypt 87*, volume 304 of *Lecture Notes in Computer Science*, pages 5–14. Springer-Verlag, 1988.
18. A. A. Kanso, *Clock-Controlled Generators*. PhD thesis, Royal Holloway and Bedford New College, University of London, Egham, London, 1999.
19. http://magma.maths.usyd.edu.au/.
20. H. Molland. Improved linear consistency attack on irregular clocked keystream generators. *FSE*, pages 109–126, 2004.
21. R. A. Rueppel. When shift registers clock themselves. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology–Eurocrypt 87*, volume 304 of *Lecture Notes in Computer Science*, pages 53–56. Springer-Verlag, 1988.
22. R. A. Rueppel. Stream ciphers. In *Gustavus J. Simmons (Ed.), Contemporary Cryptology: The Science of Information Integrity, IEEE Press*. 1992.
23. R. A. Rueppel, editor. *Analysis and Design of Stream Ciphers*. Springer-Verlag, Berlin, 1986.
24. E. Thomé. Computation of discrete logarithms in $F_{2^{607}}$. *Asiacrypt: Advances in Cryptology - Asiacrypt: International Conference on the Theory and Application of Cryptology*, pages 107–124. LNCS, Springer-Verlag, 2001.
25. D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1), pages 54–62, 1986.
26. K. Zeng, C. H. Yang and T. R. N. Rao. An improved linear syndrome algorithm in cryptanalysis with application. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology–Crypto '90*, volume 537 of *Lecture Notes in Computer Science*, pages 34–47. Springer-Verlag, 1991.
27. E. Zenner. On the efficiency of the clock control guessing attack. *ICISC*, pages 200–212, 2002.
28. M. V. Zivkovic. An algorithm for the initial state reconstruction of the clock-controlled shift register. *IEEE Transactions on Information Theory*, 37(5), page 1488, 1991.

Cache Based Power Analysis Attacks on AES

Jacques Fournier[1,2] and Michael Tunstall[3]

[1] Computer Laboratory, University of Cambridge,
JJ Thomson Avenue, Cambridge CB3 0FD, UK
[2] Gemplus Card International, Security Technologies Department,
Avenue des Jujubiers, La Ciotat, F-13705, France
jacques.fournier@gemplus.com
[3] Smart Card Centre, Information Security Group,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
m.j.tunstall@rhul.ac.uk

Abstract. This paper describes possible attacks against software implementations of AES running on processors with cache mechanisms, particularly in the case of smart cards. These attacks are based on side-channel information gained by observing cache hits and misses in the current drawn by the smart card. Two different attacks are described. The first is a combination of ideas proposed in [2] and [11] to produce an attack that only requires the manipulation of the plain text and the observation of the current. The second is an attack based on specific implementations of the **xtime** function [10]. These attacks are shown to also work against algorithms using Boolean data masking techniques as a DPA countermeasure.

1 Introduction

Several attacks have been published on using cache access events as a side-channel [2, 3, 11, 16] on DES and AES. These are predominately timing attacks taking into account the total number of cache misses in the algorithm to determine information on the secret key being used. The use of a side-channel to analyse the pattern of cache accesses is described in [12].

More recently, an attack was published using the cache lines accessed at each table look-up in the ByteSub function of an AES implemented on a PC to derive the secret key [11]. This involves detecting what cache lines are used for every table look-up to derive the secret key. This was done by having a separate process running in parallel to observe the change in the cache after each table look-up. As each process is sharing the same cache, the changes in the lines could be directly observed by the attacking process.

Attacks using the change in current signature caused by a cache miss have also been published [2] but only part of the key could be obtained. The amount of information on the key that is derived is determined by the size of the cache lines i.e. the larger the cache lines the smaller the amount of information available.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 17–28, 2006.
© Springer-Verlag Berlin Heidelberg 2006

In this paper, an extension to the attack presented in [2] is proposed. This improvement allows the entire AES key to be derived, and focuses on the cache hit event rather than the cache misses. The attack is somewhat similar to the attack described in [11], but less information is retrieved in the initial steps as the exact cache lines used are unknown. The resulting attack requires no manipulation of the cache as required in [2, 11] but involves manipulating the plain text and observing the corresponding patterns of cache hits generated. Furthermore, another attack is described based on optimisations (for performance and security reasons) used for the **xtime** function [10]. Both attacks are extended to show that these attacks are a realistic threat to DPA resistant algorithms that just use Boolean data masking to protect against DPA [6]. These attacks are described in the context of smart cards, as smart card chips are available that use a cache for data and code accesses e.g. [8, 15], and the power consumption is a readily available side-channel in smart cards.

The remainder of the paper is structured as follows: Section 2 provides some details about cache mechanisms, while Section 3 explains how the latter mechanisms influence the current to briefly describe the side-channel model. Section 4 describes the first part of the attack where roughly half of the secret key can be derived. Section 5 shows how the rest of the key can be derived by two separate methods. Section 6 illustrates how these attacks can be adapted so that they can be applied to DPA resistant algorithms. Section 8 describes some suitable countermeasures, which is followed by a conclusion.

Notation: Throughout this paper the algorithm under attack will be AES where a plaintext $P = (p_1, p_2, p_3 \dots p_{16})_{256}$ is enciphered with a secret key $K = (k_1, k_2, k_3 \dots k_{16})_{256}$. Where the subscript 256 means that the values are to this base. This notation is used throughout this paper e.g. $\texttt{F0}_{16}$ is 240 written in base 16.

2 Cache Description

The shrink of technologies along with the growing need for more sophisticated applications is currently generating a significant shift in the hardware platforms used in smart cards, which have traditionally been based on 8-bit CISC-like CPUs. More sophisticated smart cards are emerging based on 32-bit CPUs containing dedicated peripherals (cryptographic co-processors, memory managers, large memories ...) [8, 15]. Such CPUs are optimised to achieve high performance involving dedicated mechanisms that are implemented to compensate for time consuming operations or long data paths. Details about those sophisticated mechanisms can be obtained from [5, 9]. In order to understand the attacks presented in this paper, we focus on two of these mechanisms, namely pipelining and caching.

Pipelining: Pipelining is a technique whereby the execution of each instruction is decomposed into elementary and independent steps. Each step is implemented as a separate hardware block that can work in parallel. Typically, a 3-stage

pipeline can be decomposed into an Instruction Fetch (IF), an Execute (EX) stage and a Write Back (WB) stage. More sophisticated 5-stage pipelines like [9] can involve an IF stage, a DC (Decode) stage, an EX stage, a MEM (Memory access) stage and a WB stage. Each stage is designed to be completed within one clock cycle, which means that even if each instruction takes 5 clock cycles, as in the case of a 5-stage pipeline, an instruction can be issued at every clock cycle.

Caching: Smart card architectures include embedded Non-Volatile Memories (NVM) like EEPROM or Flash to store code or data. The memories usually have high read latencies where, for example, reading one byte involves reading a whole line that takes several clock cycles.This would mean that the IF stage and the MEM stage would take more than one clock cycle, which would stall the pipeline. This would considerably reduce the rate in which instructions can be issued. To compensate for these 'slow' memories, cache mechanisms are implemented. A cache is a small, fast RAM memory whose role is to buffer the lines of NVM being fetched. Due to their technology and small size (leading to faster decode and access times) caches allow a word to be fetched in one clock cycle.

When the data or instruction word is to be fetched from the NVM, the CPU will first check whether this particular word is already in the cache: if yes (this is a *cache hit*), the word is fetched directly from the cache. If, on the contrary, this particular word is not cached this is a *cache miss*. The CPU will then fetch a whole line (e.g. 16 bytes) within which the targeted word is found. This means that even if fetching this word takes more than one clock the other words of this line will already be in the cache when required.

This mechanism considerably increases the instruction issue rate and therefore performance. On Harvard architectures, the cache is applied to both the instruction and data memories in separate caches. Detailed studies of the performance enhancements of cache mechanisms can obtained from [5]. In order to keep power consumption low smart card CPUs usually only implement one level of cache with a granularity in the order of 8 to 16 bytes.

3 The Side Channel

Given the above description of the cache mechanism, we can easily see that in the case of a cache hit the pipeline is not stalled and normal execution occurs. In the case of a cache miss the pipeline flow is stalled and the NVM is accessed. In terms of side-channel information leakage (namely the power consumption) when reading data from memory:

- In the case of a cache miss, the instruction takes more cycles than a cache hit.
- In the case of a cache miss, the power consumed by the execution is significantly higher than in the case of a cache hit because NVM accesses should consume more power than a normal CPU.

With these observations we can build a power analysis attack based on the distinctive signatures of cache hits and cache misses.

The rest of the paper details a method of using this model to build an attack on AES based on cache hits and cache misses.

In our description, the first assumption is that we have a pipelined CPU embedding a one level cache mechanism for both the instructions and data. An example of a hardware simulation of this side-channel is given in [2]. To simplify our illustration, we suppose that on the architecture being attacked the NVM is accessed by lines of 16 bytes i.e. each cache miss will mean that 16 bytes are loaded into the cache.

4 The First ByteSub Function

Our attack is implemented against the AES algorithm as described in [10]. The first step of the attack targets the ByteSub function of the first round. Just before entering this function the input data is XORed with the secret key. The resulting 16 bytes enter the ByteSub function that is usually implemented as a look-up on a table of 256 entries.

4.1 The Power Consumption

An attack on this function is already described in [2]; a slightly modified version is stated here. The main difference is this attack relies purely on the observation of the side-channel described in Section 3, whereas the attack described in [2] manipulates the cache. Less information is generated but the attack is more powerful as it only needs to manipulate the messages being ciphered and observe the cache access pattern generated.

Key information can be derived from the cache access events during the table look-up depending on the order in which the look-up table is loaded into the cache. It is assumed that for each acquisition the cache has been flushed, which can easily be provoked by resetting the smart card under observation.

The first byte of the message is fixed to a value, p_1, and different values of the second byte of the message, p_2 can be tried until a cache hit occurs. At which point it is known that $p_1 \oplus k_1 \approx p_2 \oplus k_2$, which is only an approximation due to the size of the cache granularity. In the case under study (i.e. we have a cache with a granularity of 16 bytes) we can only be sure of the high nibble of the approximation given. Therefore $(p_1 \oplus p_2) \wedge \mathtt{F0}_{16}$ will give $(k_1 \oplus k_2) \wedge \mathtt{F0}_{16}$ with at most sixteen different messages i.e. all sixteen possible values for the high nibble of p_2 can be tried until a cache hit is observed.

Once $(k_1 \oplus k_2) \wedge \mathtt{F0}_{16}$ is found, $(k_2 \oplus k_3) \wedge \mathtt{F0}_{16}$ can be found using the same method by choosing p_1 and p_2 so that a cache hit is always generated between the first two look-ups, and varying p_3 until another cache hit is generated. It is important to have a cache hit between the first two look-ups, as otherwise it is not known which cache line corresponds to the observed cache hit and some information is lost. If this process is repeated for each subsequent key byte, the high nibble of each byte will be known as a function of the high nibble of the

first byte. With at most 240 acquisitions the exhaustive search to find the key of an AES implementation can be reduced from 2^{128} to 2^{68}.

In practice, this will only be true if the implementation is known. The ByteSub function can be implemented before or after the ShiftRow function, as the ShiftRow function is a bytewise permutation. A permutation is sometimes also used on the message and key on entry to the algorithm to convert the array format to the grid format used in the specification [10]. This is an optional bytewise permutation that will change addressing during the algorithm. Both permutations will change the order in which the data is treated by the ByteSub function.

In the following sections we will assume that the implementation details are known, as the added complexity due to these permutations is negligible. The grid permutation will be ignored and the ShiftRow function will be assumed to take place after the ByteSub function.

5 Finding the Rest of the Key

The first step described in Section 4 reduces the keyspace to 2^{68} and is theoretically trivial. There are two ways to continue the attack to derive the rest of the key using the same side-channel. These two independent methods are described below.

5.1 The Second ByteSub Function

The second ByteSub function (i.e. the ByteSub of the second round of AES) can be used to determine the rest of the key in a similar manner to that described in [11], and the same notation has been used for clarity. Plain texts are chosen such that there are no cache misses in the first ByteSub function, except for the first table look-up. The plain text bits that are XORed with the unknown bits of the key (i.e. the first byte and the lower nibbles of the rest of the plain text) are randomised for each acquisition. If the first look-up in the second ByteSub function is a cache hit then information on the unknown key bits can be derived. In this case the following relationship is known:

$$(2 \bullet s(p_1 \oplus k_1) \oplus 3 \bullet s(p_6 \oplus k_6) \oplus s(p_{11} \oplus k_{11}) \\ \oplus s(p_{16} \oplus k_{16}) \oplus k_1 \oplus s(k_8) \oplus 1) \wedge \mathtt{F0}_{16} = (k_1 \oplus k_2) \wedge \mathtt{F0}_{16}$$

Where the function $s(\cdot)$ represents the look-up table used in the ByteSub function and $\bullet$ represents multiplication over GF(2^8).

The value of $(k_1 \oplus k_2) \wedge \mathtt{F0}_{16}$ is known from the first part of the attack described in Section 4. The value of k_1 is unknown but given k_1 the high nibbles of k_6, k_{11}, k_{16} and k_8 can be derived. This means that there are 24 unknown bits in the equation. The evaluation of the 2^{24} possible combinations of the left hand side of the equation will be equal to $(k_1 \oplus k_2) \wedge \mathtt{F0}_{16}$ with a frequency of 1 in 16. One plaintext that produces a cache hit in the second ByteSub function will therefore reduce the unknown bits in the equation from 2^{24} to 2^{20}.

A second cache hit with a different plain text can then be analysed, the correct key values will be in the intersection of the two sets of 2^{20} values produced. With 6 evaluations of the above equation all the unknown bits can be derived. This corresponds to 96 acquisitions, as the cache hit occurs with a probability of 1/16 given that the plaintext input is mostly random. This reduces the unknown key bits from 2^{68} to 2^{44}. The cache misses could also be used as they would reduce the keyspace by 15/16, but given the small amount of acquisitions required this should not be necessary.

Any acquisition with two successive cache hits can then be used to derive information on another 5 key bytes. If the second look up in the second ByteSub function is also a cache hit, the following equation holds.

$$(2 \bullet s(p_2 \oplus k_2) \oplus 3 \bullet s(p_7 \oplus k_7) \oplus s(p_{12} \oplus k_{12}) \\ \oplus s(p_{13} \oplus k_{13}) \oplus k_2 \oplus k_1 \oplus s(k_8) \oplus 1) \wedge \mathtt{F0}_{16} = (k_1 \oplus k_2) \wedge \mathtt{F0}_{16}$$

It is faster to search through the possible values of this equation as there are 20 unknown bits, the values of k_1 and k_8 being provided by the previous step. As previously, the evaluation of this equation reduces the unknown values by a factor of 16. It is expected that 5 such equations need to be evaluated, taking the intersection as before, to provide one value for all of the key bytes in the equation. This event occurs with a probability of 1/256 so the acquisition phase will be lengthier than the previous step. A total of 1280 acquisitions should be required.

If all of the key bytes in the above equations are derived, the key can then be found by an exhaustive search of the remaining unknown key bits. This will be a search in a keyspace of size 2^{24} (i.e. 9 complete key bytes are given by the formulae above, for the remaining six the high nibble is known, leaving 24 unknown bits), which can easily be exhausted on a PC. This is fortunate as continuing the attack for three successive cache hits would be difficult as the probability of seeing such an event is 1/4096, which would make the attack excessively time consuming.

The last set of equation evaluations are time consuming, which means that it can be advantageous to acquire less data and let the exhaustive search complete the key search. If, for example, an attacker takes 768 acquisitions the expected exhaustive key search would be around 2^{32}. Another means of speeding up the evaluation of the possible key values for the second equation would be to use the event of a cache hit followed by a cache miss, which occurs with a probability of 15/256, each of which will reduce the unknown keyspace by 15/16.

5.2 The xtime Function

A second method to reduce the key search space is to focus on the **xtime** function. The **xtime** function is a multiplication by 2 over GF(2^8) and is used in the MixColumn function [10]. The **xtime** function is a bit shift followed by a conditional XOR (as shown in Algorithm 1). This is difficult to implement securely in smart cards as there is a danger that the result of the conditional test can be

Algorithm 1. The **xtime** function

Input: $x = (x_7, x_6, \ldots, x_0)_2$
Output: $y = \mathbf{xtime}(x)$

$y \leftarrow (x << 1) \wedge \mathrm{FF}_{16}$
if $x_7 = 1$ **then**
  $y \leftarrow y \oplus 1\mathrm{B}_{16}$
end
return y

leaked through the power consumption as the two branches will take different amounts of time to complete. Even if this is implemented so that the calculation always takes the same amount of time, there is still a risk of a partitioning attack [14].

In smart cards a possible replacement for this function is with a look-up table of 256 bytes to avoid any conditional testing. This protects the implementation against Simple Power Analysis but the table will be in Non-Volatile Memory so will be accessed via the cache as with the look-up table used in the ByteSub function. The pattern of cache hits and misses can therefore be analysed in a similar way to the first phase of the attack described in Section 4. The first look-up to the **xtime** table will be a cache miss, if this is followed by a cache hit then:

$$s(p_1 \oplus k_1) \wedge \mathrm{F0}_{16} = s(p_6 \oplus k_6) \wedge \mathrm{F0}_{16}$$

Where, as previously, the $s(.)$ represents the look-up table in the ByteSub function. The right hand side of the equation uses $p_6 \oplus k_6$ rather than $p_4 \oplus k_4$, that would be expected due to the ShiftRow function, due to the ShiftRow function. In this equation there are 2^{12} possible combinations given that the high nibble of k_6 is known as a function of k_1, from the first part of the attack described in Section 4. Searching through all the combinations will give 2^8 possible values for the pair (k_1, k_6). Due to the non-linear nature of the $s(\cdot)$ function another cache hit can be found with a different message that will provide a different set of 2^8 values. The correct key will be in the intersection between the two sets of possible values. After three cache hits with three different messages are found there should only be one hypothesis for both k_1 and k_6. Each cache hit will occur with a probability of $1/16$, so 48 acquisitions should be enough to find the value of k_1 and k_6.

The next cache access is the first **xtime** function call for the next output byte. The values for p_1 and p_6 can be fixed so that a cache hit is always generated between the first two **xtime** look-ups. If a cache hit occurs for the next **xtime** look-up then:

$$s(p_6 \oplus k_6) \wedge \mathrm{F0}_{16} = s(p_2 \oplus k_2) \wedge \mathrm{F0}_{16}$$

In this case k_6 is known and the high nibble of k_2 is known, as k_1 has been determined the high nibble of all the key bytes are known. The 4 unknown bits of k_2 in the equation can be exhausted for the value of p_2 that provokes a cache hit. One cache hit of this nature would be enough to determine the 4 unknown bits. This process can be continued with the following equations:

$$s(p_2 \oplus k_2) \wedge \mathrm{F0}_{16} = s(p_7 \oplus k_7) \wedge \mathrm{F0}_{16}$$
$$s(p_7 \oplus k_7) \wedge \mathrm{F0}_{16} = s(p_3 \oplus k_3) \wedge \mathrm{F0}_{16}$$
$$s(p_3 \oplus k_3) \wedge \mathrm{F0}_{16} = s(p_8 \oplus k_8) \wedge \mathrm{F0}_{16}$$
$$s(p_8 \oplus k_8) \wedge \mathrm{F0}_{16} = s(p_4 \oplus k_4) \wedge \mathrm{F0}_{16}$$
$$s(p_4 \oplus k_4) \wedge \mathrm{F0}_{16} = s(p_5 \oplus k_5) \wedge \mathrm{F0}_{16}$$

This can determine the first 8 bytes of the key with 192 acquisitions, leaving an exhaustive search of 2^{32} possible keys. An exhaustive search of 2^{32} is prohibitive so further analysis would be advantageous.

The next 4 cache accesses can be analysed requiring a further 64 acquisitions (for a total of 256 acquisitions) and reduces the amount of unknown key bits to 16. As an exhaustive search of 2^{16} is trivial, no further acquisitions are required to derive the key.

6 Application to DPA Resistant Implementations

In smart cards implementations of cryptographic algorithms like AES are implemented with countermeasures to protect against Differential Power Analysis (DPA) [6]. One of the techniques used to protect the AES is by masking the data being manipulated with a random value. The data is then manipulated in such a way that the value present in memory is always masked with the same random. This mask is then removed at the end of the algorithm to produce the ciphertext. The most common form of masking is Boolean masking where all data manipulated is treated after being XORed with a random, such that the result is also XORed with the same random value. An example of this sort of implementation can be found in [1].

The size of the random is generally limited as look-up tables need to be randomised before the execution of the algorithm so that the input and output values of the s-box leak no information. An example of how this is done is given in Algorithm 2. As illustrated in the latter, the random used for masking the input data can be no larger than n, and the random used for the output value can be no larger that x.

In the case of AES both $\mathbf{R}$ and r are on one byte, which means that the random mask during the calculation of AES will also be on one byte.

Algorithm 2. Randomising S-Box Values

Input: $S = (s_0, s_1, s_2, \ldots, s_n)_x$ containing the s-box, $\mathbf{R}$ a random $\in [0, n]$, and r a random $\in [0, x)$.
Output: $RS = (rs_0, rs_1, rs_2, \ldots, rs_n)_x$ containing the randomised s-box.

for $i \leftarrow 0$ **to** n **do**
  $rs_i \leftarrow s_{(i \oplus \mathbf{R})} \oplus r$
end
return RS

6.1 Implementing the Attack

The described attack can be implemented as described in the above sections, as the random will provide one byte of variation. In all the equations used to test key hypotheses, the values generated are always compared with the neighbouring byte. If, for example, all bytes in the algorithm are masked with the random **R** the first phase of the attack described in Section 4 will give $(k_1 \oplus \mathbf{R} \oplus k_2 \oplus \mathbf{R}) \wedge \mathtt{F0}_{16}$. The **R**'s will cancel leaving $(k_1 \oplus k_2) \wedge \mathtt{F0}_{16}$ as with the approach detailed in Section 4. The random will just change the order of the cache lines and the order of the bytes within them, but the same plaintext values will give the same cache access pattern.

This does not mean that a DPA resistant algorithm is as easy to attack as a naive implementation. There will be an initialisation phase during the algorithm execution where the look-up table for the ByteSub function is randomised and written into RAM, as described in Algorithm 2. In order for the cache to reveal information as described above, enough time needs to have passed between the execution of Algorithm 2 and the ciphering algorithm so that the cache no longer contains the randomised look-up table. In theory, it may be possible to apply the attack in [3] but it is necessary to know the cache lines that no longer contain the randomised look-up table.

6.2 The xtime Function

The attack described in Sections 4 and 5 can work against a DPA resistant algorithm assuming the randomised look-up table is no longer present in the cache, but this assumption is probably not reasonable. It would be simpler to directly attack the **xtime** function instead of the ByteSub function. The **xtime** function has the property that if $y = \mathbf{xtime}(x)$ then $y \oplus \mathbf{R} = \mathbf{xtime}(x \oplus \mathbf{R})$ for $\mathbf{R} \in [0, 255]$ i.e. the data mask will carry across the **xtime** function. This means that there is no need to load the **xtime** table into RAM in a DPA resistant implementation of AES.

In this case the attack described in Section 5.2 can be extended to recover all of the key data rather than just the first byte and the lower nibbles. The first equation for a cache hit between the first and second **xtime** look-up becomes:

$$(s(p_1 \oplus k_1) \oplus \mathbf{R}) \wedge \mathrm{F0}_{16} = (s(p_6 \oplus k_6) \oplus \mathbf{R}) \wedge \mathrm{F0}_{16}$$
$$s(p_1 \oplus k_1) \wedge \mathrm{F0}_{16} = s(p_6 \oplus k_6) \wedge \mathrm{F0}_{16}$$

In this case there are 16 unknown bits and an evaluation will reduce the keyspace by a factor of 16. After four evaluations of this equation a single solution can be found for the pair (k_1, k_6). This cache hit event occurs with a probability of 1/16 for a random plain text. An attack therefore requires a maximum of 64 acquisitions before being able to derive the key byte.

The attack can continue in the same manner as the attack described in Section 5.2 but the total attack will require around 480 acquisitions and an exhaustive search of 2^{16} to derive the entire key.

Algorithm 3. Calculating **xtime** on a 32 bit platform

Input: $A = (a_0, a_1, a_2, \ldots, a_{15})_{256}$.
Output: $B = (b_0, b_1, b_2, \ldots, b_{15})_{256}$.

for $i \leftarrow 0$ **to** 3 **do**
  LOAD $R_1 \leftarrow (a_{4i}, a_{4i+1}, a_{4i+2}, a_{4i+3})$
  $R_2 \leftarrow R_1 \wedge 80808080_{16}$
  $R_2 \leftarrow R_2 >> 7$
  $R_3 \leftarrow R_2 * 1\mathrm{B}_{16}$
  $R_1 \leftarrow R_1 << 1$
  $R_1 \leftarrow R_1 \wedge \mathrm{FEFEFEFE}_{16}$
  $R_1 \leftarrow R_1 \oplus R_3$
  STORE $(b_{4i}, b_{4i+1}, b_{4i+2}, b_{4i+3}) \leftarrow R_1$
end

return B

7 Countermeasures

Several countermeasures can provide a protection against this attack in smart cards. A more complete discussion of the countermeasures for protecting algorithms against attacks using a side-channel to observe cache accesses is given in [13]. A summary of these countermeasures is presented here:

Programming Instructions: On some architectures the caching of data can be avoided by fetching data without caching it. Such instructions do incur performance penalties but they have the advantage of always taking the same amount of time to execute.

Random Delay: The use of dummy code in cryptographic algorithms is a common countermeasure used to prevent side-channel attacks. Such mechanisms lower the signal-to-noise ratios of such side-channels, thus adding another level of difficulty to the implementation of this attack. A discussion of this effect is given in [4], further discussion in the specific context of side-channel attacks on cache access patterns appears in [13].

Random Order: If all the functions are conducted in a random order it will not be possible to determine any relationship between a cache hit/miss and the actual values being manipulated, which can either be implemented in hardware [13] or software [7]. In an actual DPA resistant implementation this countermeasure would be expected, as it renders power attacks exceedingly difficult especially when combined with data masking.

Calculating the xtime Function: On a 32-bit architecture, the **xtime** operation can be computed without a performance penalty compared to the table look-up implementation. On an assembly instruction level, the table look-up implementation of the **xtime** could be implemented simply with four instructions, i.e. two LOAD, one addition a STORE function.

On a 32-bit architecture Algorithm 3 can be implemented, which not only avoids any memory accesses but may be faster on a 32-bit platform as the

operation would take 8 instruction cycles, but calculates 4 bytes in parallel. The side-channel issues concerning the visibility of the most significant bit of each byte is less of an issue as four bytes are being manipulated separately.

8 Conclusion

In this paper we propose an attack against software AES implemented on a smart card with cache mechanisms. Our attack is based on the observation of the power consumption information leakage generated by the different mechanisms behind the caching techniques. We first explain how cache events generate different side-channel signatures, before showing how varying the input message on the first round can be combined with this observation to reduce the AES key search space from 2^{128} to 2^{68}.

We propose two alternatives to find the remaining key bits either by focussing on cache events during a ByteSub operation of the second AES round, or by targeting the **xtime** of the MixColumn operation in the first round. Furthermore, we argue that these attacks are also valid against implementations where masking techniques are implemented as a countermeasure against DPA-like attacks.

This shows that when implementing cryptography on a given processor, the specificities of the processor must be taken into account in order to have a secure implementation. Caches are highly important features in high performance embedded processors but they need to be carefully used when executing cryptographic algorithms like AES.

References

1. M.-L. Akkar and C. Giraud. An implementation of DES and AES secure against some attacks. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptogaphic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer-Verlag, 2001.
2. D. J. Bernstein. Cache timing attacks on AES, 2004. `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`.
3. G. Bertoni, V. Zaccaria, L. Breveglieri, M. Monchiero, and G. Palermo. AES power attack based on induced cache miss and countermeasures. In *International Symposium on Information Technology: Coding and Computing – ITCC 2005*, pages 586–591. IEEE Computer Society, 2005.
4. C. Clavier, J.-S. Coron, and N. Dabbous. Differential power analysis in the presence of hardware countermeasures. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer-Verlag, 2000.
5. J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 2003.
6. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
7. T. S. Messerges. *Power Analysis Attacks and Countermeasures for Cryptographic Algorithms*. PhD thesis, University of Illinois, Chicago, 2000.

8. MIPS-Technologies. SmartMIPS ASE. `http://www.mips.com/content/Products/`.
9. MIPS-Technologies. MIPS^{TM}architecture for programmers volume I: Introduction to the MIPS32^{TM}architecture. Technical Report MD00082, Revision 0.95, March 2001.
10. National Institute of Standards and Technology. Advanced encryption standard (AES) (FIPS–197), 2001.
11. D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: the case of AES. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 2006.
12. D. Page. Theoretical use of cache memory as a cryptanalytic side–channel. Cryptology ePrint Archive, Report 2002/169, 2002. `http://eprint.iacr.org/`.
13. D. Page. Defending against cache based side-channel attacks. *Information Security Technical Report*, 8(1):30–44, April 2003.
14. J. R. Rao, P. Rohatgi, H. Scherzer, and S. Tinguely. Partitioning attacks: or how to rapidly clone some gsm cards. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 31–41, 2002.
15. Infineon Technologies AG Secure and Mobile Solutions Security Group. Security & chip cards ICs SLE88Cx4000P, preliminary short product information 04.03, 2003.
16. Y. Tsunoo, T. Saito, T. Suzaki, M. Shigeri, and H. Miyauchi. Cryptanalysis of DES implemented on computers with cache. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 62–76. Springer-Verlag, 2003.

Distinguishing Attack on SOBER-128 with Linear Masking

Joo Yeon Cho and Josef Pieprzyk

Centre for Advanced Computing – Algorithms and Cryptography,
Department of Computing, Macquarie University,
NSW, Australia, 2109
{jcho, josef}@ics.mq.edu.au

Abstract. We present a distinguishing attack against SOBER-128 with linear masking. We found a linear approximation which has a bias of $2^{-8.8}$ for the non-linear filter. The attack applies the observation made by Ekdahl and Johansson that there is a sequence of clocks for which the linear combination of some states vanishes. This linear dependency allows that the linear masking method can be applied. We also show that the bias of the distinguisher can be improved (or estimated more precisely) by considering quadratic terms of the approximation. The probability bias of the quadratic approximation used in the distinguisher is estimated to be equal to $O(2^{-51.8})$, so that we claim that SOBER-128 is distinguishable from truly random cipher by observing $O(2^{103.6})$ keystream words.

Keywords: Distinguishing attack, Stream ciphers, Linear masking, Modular addition, SOBER-128.

1 Introduction

One of the recent trends in designing stream ciphers is that stream ciphers are word-oriented. Since the operation of ciphers are based on words and keystreams are produced word by word at each clock, they are fast and efficient when implemented in software. This class of ciphers includes SNOW [3], SOBER [4], MUGI [2] and many others. In particular, among eSTREAM stream cipher submissions, the word-oriented ciphers are Dragon, Phelix, NLS, HC-256 to mention a few [1].

The SOBER-128 is one of recently proposed word-oriented stream ciphers. The cipher is built using the classical structure with a linear feedback shift register (LFSR) and a non-linear filter function. SOBER-128 is an improved version of SOBER-t32 which was a candidate of the stream cipher primitives in NESSIE project [10]. The non-linear function has been strengthened by adding a fixed rotation and the second S-box transformation. The stuttering phase that was present in SOBER-t32 is not used in SOBER-128.

In this work, we develop a distinguishing attack on SOBER-128 with linear masking introduced by Coppersmith, Halevi and Jutla at CRYPTO 2002 [6]. The authors of [6] study two types of distinguishing characteristic of non-linear processes : the linear approximation and the low diffusion. We use the linear approximation to develop the attack against SOBER-128. In addition, we combine

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 29–39, 2006.
© Springer-Verlag Berlin Heidelberg 2006

a quadratic polynomial with the linear approximations for a precise estimation of the expected probability bias.

The authors of [6] shows that if there is a linear approximation σ of the non-linear function with bias ϵ, then the bit $\xi_j = \bigoplus_{j \in J} \sigma_j$ has the bias of $\epsilon^{|J|}$, where J is a set of steps such that $\bigoplus_{j \in J} s_j = 0$, provided s_j is a state bit of a linear feedback shift register.We claim that the bias of ξ_j could be slightly higher than $\epsilon^{|J|}$ when quadratic terms are considered.

Our attack on SOBER-128 is based on two linear approximations that exhibit a big enough probability bias. We observe that the bias of the quadratic approximation for non-linear filter of SOBER-128 is $O(2^{-51.8})$. Therefore, we claim that SOBER-128 is distinguishable from a random process by observing around $O(2^{103.6})$ keystream words.

This paper is organized as follows. In Section 2, the distinguishing attack with linear masking using a linear approximation is briefly described. In Section 3, the structure of SOBER-128 is given. In Section 4, we derive linear approximations on the nonlinear Filter (NLF). In Section 5, a linear distinguishing attack is applied by using the derived approximation. Section 6 applies an improved distinguishing attack using a quadratic approximation. Conclusions are given in Section 7.

2 Linear Masking Using Linear Approximation

We describe briefly the linear masking method for the linear attack which is presented in [6]. The attack is applicable for a class of stream ciphers with a special structure that consist of the linear process (LF) and the non-linear process (NF). The state in a such cipher is identified by a pair: linear state x and non-linear state z. The cipher works in steps (clocks) and at each step i, the cipher

- sets the linear state as $x_i := LF(x_{i-1})$,
- calculates two variables $u_i := L1(x_i)$ and $v_i := L2(x_i)$, where $L1, L2$ are linear functions,
- determines non-linear state $z_i := NF(z_{i-1} \oplus u_i) \oplus v_i$,
- outputs z_i.

Assume that we have a linear function $l : \{0,1\}^{2n} \to \{0,1\}$, such that

$$Pr[l(z, NF(z)) = 0] = \frac{1}{2}(1 + \epsilon), \quad |\epsilon| > 0^1$$

in other words, the function l is a linear approximation of the non-linear function NF and ϵ is the bias of the approximation.

Suppose that the adversary observes a bit $\sigma_j = l(z_j \oplus u_j, NF(z_j) \oplus v_j)$ where the variables u and v come from a linear space. Then there is always a linear combination of steps (not necessarily consecutive) for which the variables u and v vanish. Let J be a set of such steps for which $\bigoplus_{j \in J} u_j = \bigoplus_{j \in J} v_j = 0$. Thus, we can write

[1] This definition simplifies the computation of bias of multiple approximations when the piling-up lemma is considered. If we have n independent approximations, the probability of n approximations becomes $\frac{1}{2}(1 + \epsilon^n)$. Whereas, if p is defined by a form of $p = \frac{1}{2} + \epsilon$, the probability of n approximations becomes $\frac{1}{2}(1 + 2^{n-1}\epsilon^n)$.

$$\begin{aligned} &\bigoplus_{j\in J}\sigma_j = \bigoplus_{j\in J} l(z_j, NF(z_j)) \oplus \bigoplus_{j\in J} l(u_j + v_j) \\ &= \bigoplus_{j\in J} l(z_j, NF(z_j)) \end{aligned} \tag{1}$$

Therefore, if the number of elements in the set J is n, $\bigoplus_{j\in J}\sigma_j$ has the bias of ϵ^n.

Using this bias, an adversary can reliably distinguish the stream cipher from the random process by observing around ϵ^{-2n} outputs. For more details, see [6].

3 Brief Description of SOBER-128

The SOBER-128 consists of a linear feedback shift register (LFSR) and a nonlinear filter (NLF). The LFSR consists of 17 words state registers which is denoted by the vector $(s_t, \cdots, s_{t+16})$. Since each s_i is a 32-bit integer, the size of LFSR is 544 bits. The new state of the LFSR is generated by the following connection polynomial

$$s_{t+17} = s_{t+15} \oplus s_{t+4} \oplus \gamma s_t,$$

where γ = 0x00000100 (hexadecimal).

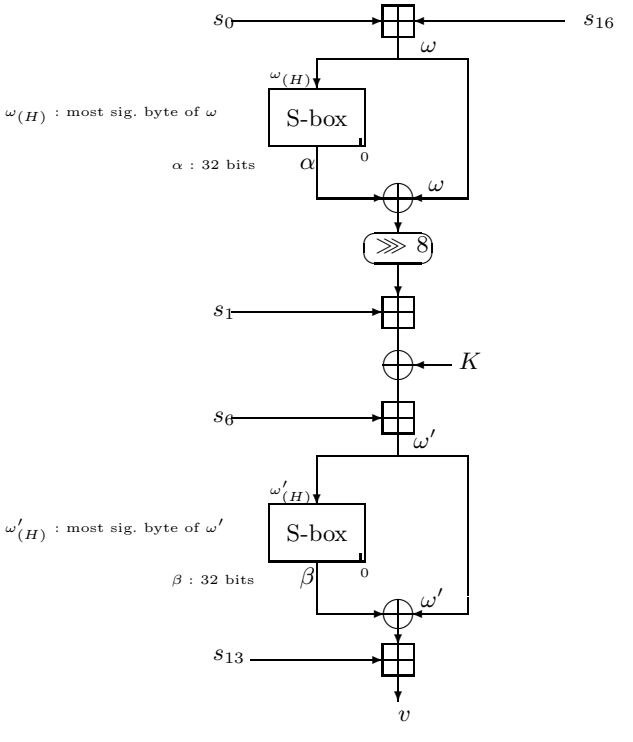


Fig. 1. The non-linear filter (NLF) of SOBER-128

A Nonlinear Filter (NLF) produces an output word z_t by taking $s_t, s_{t+1}, s_{t+6}, s_{t+13}, s_{t+16}$ from the LFSR states and the constant K. The NLF consists of two substitution functions (S-box), one rotation, four adders modulo 2^{32} and three XOR additions. For the detail description of the NLF, see Figure 1.

The K is a 32-bit key-dependent constant. The function f is defined as $f(a) = \text{S-box}(a_H) \oplus a$ where the S-box is 8×32-bit and a_H is the most significant 8 bits of 32-bit word a. The output z_t of the nonlinear filter is described as

$$z_t = f((((f(s_t \boxplus s_{t+16}) \ggg 8) \boxplus s_{t+1}) \oplus K) \boxplus s_{t+6}) \boxplus s_{t+13},$$

where $\boxplus$ denotes an addition modulo 2^{32} and $\ggg 8$ denotes a 8-bit right rotation. The LFSR states and a constant K are initialized from the 128-bit secret key using the initialization procedure. More details can be found in [8].

4 Deriving Linear Approximations on NLF

According to the structure of the non-linear filter, the following equation holds for the least significant bit (see Figure 1). Let us denote that α is 32-bit output of the first S-box, β is 32-bit output of the second S-box and ω is 32-bit output of the addition of s_0 and s_{16}, respectively. Then, the following equation holds at any clock

$$\alpha_{(8)} \oplus \beta_{(0)} \oplus \omega_{(8)} \oplus s_{1,(0)} \oplus s_{6,(0)} \oplus s_{13,(0)} \oplus K_{(0)} = z_{(0)}, \tag{2}$$

where $x_{t,(i)}$ stands for the i-th bit of the 32-bit word x at clock t. (This notation will be also used for the other equations.)

We will find the best linear approximation for $\alpha_{(8)}$, $\beta_{(0)}$ and $\omega_{(8)}$. In order to apply the linear masking method for a distinguisher of SOBER-128, we use a low weight linear relationship among the states of LFSR which was presented for attack on SOBER-t32 [7]. The LFSR of SOBER-128 is not same as that of SOBER-t32 but the following relationship still holds for both stream ciphers

$$s_{t+\tau_1} \oplus s_{t+\tau_2} \oplus s_{t+\tau_3} \oplus s_{t+\tau_4} \oplus s_{t+\tau_5} \oplus s_{t+\tau_6} = 0 \tag{3}$$

with $\tau_1 = 0, \tau_2 = 11, \tau_3 = 13, \tau_4 = 4 \cdot 2^{32} - 4, \tau_5 = 15 \cdot 2^{32} - 4, \tau_6 = 17 \cdot 2^{32} - 4$. This linear recurrence is valid for each bit position individually.

4.1 Linear Approximations of $\alpha_{(8)}$

The bit $\alpha_{(8)}$ is the 8-th output bit of the first S-box. The input of the S-box is the most significant 8-bit of the addition of the state register s_0 and s_{16}. Thus, $\alpha_{(8)}$ is completely determined by both s_0 and s_{16} registers. However, the input of the S-box is mostly affected by the most significant 8 bits of the register s_0 (which is called $s_{0,(H)}$) and s_{16} (which is called $s_{16,(H)}$), respectively. Hence, we try to find the best linear approximation for $\alpha_{(8)}$ from the whole set of linear combinations of $s_{0,(H)}$ and $s_{16,(H)}$. In order to calculate the correlation of each combination, we introduce the carry bit $carry_1$, which is induced from the addition of two 24

least significant bits of s_0 and s_{16}. We regard the bit $carry_1$ as a uniform and independent variable. Then,

$$\text{The input of the first S-box} = s_{0,(H)} \boxplus s_{16,(H)} \boxplus carry_1$$

We build the truth table with 2^{17} rows and 2^{16} columns. Each row corresponds to the unique collection of input variables (8 bits of $s_{0,(H)}$, 8 bits of $s_{16,(H)}$, and a single bit for $carry_1$). Each column relates to the unique linear combination of bits from $s_{0,(H)}$ and $s_{16,(H)}$. In result, we have found four linear approximation for $\alpha_{(8)}$, which have the best bias (see Table 1).

Table 1. The best linear approximations for $\alpha_{(8)}$

linear approximation	bias
$s_{0,(25)} \oplus s_{0,(26)} \oplus s_{0,(28)} \oplus s_{0,(29)} \oplus s_{16,(26)} \oplus s_{16,(29)}$	1/2(1 - 0.057618)
$s_{0,(26)} \oplus s_{0,(29)} \oplus s_{16,(25)} \oplus s_{16,(26)} \oplus s_{16,(28)} \oplus s_{16,(29)}$	1/2(1 - 0.057618)
$s_{0,(26)} \oplus s_{0,(28)} \oplus s_{0,(29)} \oplus s_{16,(25)} \oplus s_{16,(26)} \oplus s_{16,(29)}$	1/2(1 - 0.057618)
$s_{0,(25)} \oplus s_{0,(26)} \oplus s_{0,(29)} \oplus s_{16,(26)} \oplus s_{16,(28)} \oplus s_{16,(29)}$	1/2(1 - 0.057618)

Let us choose the first approximation from the table so

$$\alpha_{(8)} \approx s_{0,(25)} \oplus s_{0,(26)} \oplus s_{0,(28)} \oplus s_{0,(29)} \oplus s_{16,(26)} \oplus s_{16,(29)} \tag{4}$$

and the probability is $\frac{1}{2}(1 - 0.057618) = \frac{1}{2}(1 - 2^{-4.1})$.

4.2 Linear Approximation of $\beta_{(0)}$

The best linear approximation of $\beta_{(0)}$ can be obtained by a similar approach we have applied for $\alpha_{(8)}$ with an addition trick.

The S-box of the SOBER-128 consists of two different S-boxes which are the Skipjack S-box and the S-box that was custom-designed by the researchers from QUT. Using the structure of S-Box, we can observe that not only the input of the second S-box but also the 8-bit output of the S-box determines the bit β_0 completely. The output of the Skipjack S-box is the most significant 8 bits of the subtraction of the state register s_{13} from the output z. Thus, $\beta_{(0)}$ is determined by both s_{13} and z. However, in a similar way to $\alpha_{(8)}$, the most significant 8 bits of the register s_{13} (which is called $s_{13,(H)}$) and the output z (which is called $z_{(H)}$) contribute to β_0. Hence, we try to find the best linear approximation for $\beta_{(0)}$ from the whole set of linear combinations of $s_{13,(H)}$ and $z_{(H)}$.

In order to calculate the best linear approximation, we also introduce the carry bit $carry_2$ which is induced from the addition of two 24 least significant bits of the register s_{13} and the output of the second f-function. We regard the bit $carry_2$ as a uniform and independent variable. So,

$$\text{The output of the Skipjack S-box} \boxplus s_{13,(H)} \boxplus carry_2 = z_{(H)}$$

In a similar way to $\alpha_{(8)}$, we build the truth table with 2^{17} rows and 2^{16} columns for $\beta_{(0)}$. Each row corresponds to the collection of variables (8 bits of $s_{13,(H)}$, 8-bit output of the Skipjack S-box , and a single bit for $carry_2$). Each column relates to the unique linear combination of bits from $s_{13,(H)}$ and $z_{(H)}$.

Table 2 displays the best and the second best linear approximations of $\beta_{(0)}$.

Table 2. Linear approximations on $\beta_{(0)}$

linear approximation	bias
$s_{13,(29)} \oplus s_{13,(30)} \oplus z_{(29)} \oplus z_{(30)}$	1/2(1+0.07666)
$s_{13,(31)} \oplus z_{(31)}$	1/2(1+0.072388)
$s_{13,(30)} \oplus s_{13,(31)} \oplus z_{(30)} \oplus z_{(31)}$	1/2(1+0.072388)

Hence, the best linear approximation on $\beta_{(0)}$ is such that

$$\beta_{(0)} \approx s_{13,(29)} \oplus s_{13,(30)} \oplus z_{(29)} \oplus z_{(30)} \tag{5}$$

with the probability of $\frac{1}{2}(1 + 0.07666) = \frac{1}{2}(1 + 2^{-3.7})$.

Remark. We may improve the bias by considering non-linear approximations for $\beta_{(0)}$ in such a way that the approximations take the following form.

$$\beta_{(0)} = \text{linear}(s_{13,(H)}) \oplus \text{nonlinear}(z_{(H)})$$

Since only linear$(s_{13,(H)})$ vanishes by the linear masking method and nonlinear $(z_{(H)})$ becomes a part of a distinguisher, we may improve the bias by manipulating all the non-linear monomials which are generated by the 8 bits of $z_{(H)}$.

4.3 Linear Approximations of $\omega_{(8)}$

The bit $\omega_{(8)}$ is the 8-th bit of output which is produced by adding the registers s_0 and s_{16}. Clearly $\omega_{(8)}$ is determined by the least significant 9 bits of s_0 and and s_{16} (which are denoted as $s_{0,(L)}$ and $s_{16,(L)}$ respectively). Thus,

$$\omega_{(8)} = (s_{0,(L)} \boxplus s_{16,(L)})_{(8)} \tag{6}$$

In order to find the best approximation for $\omega_{(8)}$, a truth table is constructed by considering all the possible linear combinations among the bit string $s_{0,(L)}$

Table 3. The best linear approximations for $\omega_{(8)}$

linear approximation	bias
$s_{0,(8)} \oplus s_{0,(7)} \oplus s_{16,(8)}$	1/2(1+0.5)
$s_{0,(8)} \oplus s_{16,(8)} \oplus s_{16,(7)}$	1/2(1+0.5)
$s_{0,(8)} \oplus s_{0,(7)} \oplus s_{0,(0)} \oplus s_{16,(8)} \oplus s_{16,(0)}$	1/2(1+0.5)
$s_{0,(8)} \oplus s_{0,(0)} \oplus s_{16,(8)} \oplus s_{16,(7)} \oplus s_{16,(0)}$	1/2(1+0.5)

and $s_{16,(L)}$. In result, we found the four best linear approximations for $\omega_{(8)}$ with same bias (see Table 3). Let us choose the first approximation from the table. Then,

$$\omega_{(8)} \approx s_{0,(8)} \oplus s_{16,(8)} \oplus s_{0,(7)} \tag{7}$$

and the probability of approximation is $\frac{1}{2}(1 + 2^{-1})$.

5 Distinguishing Attack on SOBER-128 with Linear Masking

Recall Equation (2) on NLF. If we replace $\alpha_{(8)}$, $\beta_{(0)}$ and $\omega_{(8)}$ by Approximations (4), (5) and (7) respectively, we build a linear approximation on NLF as follows.

$$\underbrace{s_{0,(25)} \oplus s_{0,(26)} \oplus s_{0,(28)} \oplus s_{0,(29)} \oplus s_{16,(26)} \oplus s_{16,(29)}}_{\alpha_{(8)}} \oplus \underbrace{s_{13,(29)} \oplus s_{13,(30)} \oplus z_{(29)} \oplus z_{(30)}}_{\beta_{(0)}}$$
$$\oplus \underbrace{s_{0,(8)} \oplus s_{16,(8)} \oplus s_{0,(7)}}_{\omega_{(8)}} \oplus s_{1,(0)} \oplus s_{6,(0)} \oplus s_{13,(0)} \oplus K_{(0)} = z_{(0)} \tag{8}$$

where the bias is

$$\frac{1}{2}(1 + 2^{-4.1} \cdot 2^{-3.7} \cdot 2^{-1}) = \frac{1}{2}(1 + 2^{-8.8}) \tag{9}$$

Let us divide Approximation (8) into two parts : a linear combination of the state bits and that of the output bits. Then, Approximation (8) will be

$$\begin{aligned} &s_{0,(25)} \oplus s_{0,(26)} \oplus s_{0,(28)} \oplus s_{0,(29)} \oplus s_{16,(26)} \oplus s_{16,(29)} \oplus s_{13,(29)} \oplus s_{13,(30)} \\ &\oplus s_{0,(8)} \oplus s_{16,(8)} \oplus s_{0,(7)} \oplus s_{1,(0)} \oplus s_{6,(0)} \oplus s_{13,(0)} \oplus K_{(0)} \\ &= z_{(0)} \oplus z_{(29)} \oplus z_{(30)} \end{aligned} \tag{10}$$

If we apply the linear masking method described in Section 2, then, the left part of Approximation (10) with linear masking vanishes by the linear connection of Equation (3).

Therefore, we can build a distinguisher of $\bigoplus_{t=\tau_1}^{\tau_6}(z_{(0)} \oplus z_{(29)} \oplus z_{(30)})$ with the bias of $(\mathbf{2^{-8.8}})^{\mathbf{6}} = \mathbf{2^{-52.8}}$.

6 An Improved Distinguishing Attack on SOBER-128

In this section, we improve the bias of the distinguisher by introducing an idea of quadratic approximations with linear masking. This idea is applied to the approximation of $\omega_{(8)}$. We show that the bit $\omega_{(8)}$ with linear masking is $(2^{-1})^5$ rather than $(2^{-1})^6$.

This section is organized as follows. First, we derive a general formula for the bias of a quadratic monomial with linear masking. Then, the formula is applied to the modular addition which is the case of $\omega_{(8)}$.

6.1 Correlation of Quadratic Monomials

Let us assume that a connection polynomial of LFSR has the weight n. That is, $\bigoplus_{i=1}^{n} x_i = 0$ where x_i represents one bit of the state register. Then, the weight of the vector $\rho = (x_1, x_2, \ldots, x_n)$ is always even. This means that one of the component of ρ is completely determined by the others. In general, the space of ρ is 2^{n-1}.

If we consider a monomial of degree d such that $\sigma_d = x_{i_1} x_{i_2} \cdots x_{i_d}$, then, the monomial σ_d is correlated due to the restriction on the space of ρ. It is clear that such correlation always exists and is dependent on the degree d and the weight n.

Let us consider a quadratic monomial which is the simplest form of non-linear function.

Lemma 1. *Given two vectors* $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ *and* $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$ *such that* $\bigoplus_{i=1}^{n} x_i = 0$ *and* $\bigoplus_{i=1}^{n} y_i = 0$, *then* $\sigma_{xy} = \bigoplus_{i=1}^{n} x_i y_i$ *is a Boolean function in* $GF(2^{2n}) \rightarrow GF(2)$ *with the bias determined by the following probability*

$$Pr[\sigma_{xy} = 0] = \begin{cases} \frac{1}{2}(1 + 2^{-n+2}) & n \text{ is even} \\ \frac{1}{2}(1 + 2^{-n+1}) & n \text{ is odd} \end{cases} \tag{11}$$

Proof. *We count how many times the zero (or one) happens when all the possible values of two vectors* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are considered.*

At first, let us consider when n *is even. Assuming that* $x_1 = \cdots = x_n = 0$. *Then,* σ_{xy} *is always zero for all values of* $\boldsymbol{y}$. *Thus, the zero count is* 2^{n-1}.

Secondly, let assume that $x_1 = \cdots = x_n = 1$. *Then,* σ_{xy} *is again always zero for all values of* $\boldsymbol{y}$ *because the weight of* $\boldsymbol{y}$ *is even. Thus, the zero count increases* 2^{n-1}.

In other values of $\boldsymbol{x}$, *the number of one is equal to that of zero. Thus, the zero count increases* $2^{n-2} \cdot (2^{n-1} - 2) = 2^{n-1} \cdot (2^{n-2} - 1)$.

All together, the zero count becomes $2^{n-1} + 2^{n-1} + 2^{n-1} \cdot (2^{n-2} - 1)$. *Therefore, the correlation becomes*

$$\frac{2^{n-1} + 2^{n-1} + 2^{n-1} \cdot (2^{n-2} - 1)}{2^{n-1} \cdot 2^{n-1}} = \frac{1 + 1 + 2^{n-2} - 1}{2^{n-1}} = \frac{1}{2}(1 + 2^{-n+2})$$

A proof is similar when n *is odd.* □

The following corollary is useful when a combined monomial with an output bit is considered.

Corollary 1. *If the vector* $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ *satisfies the condition* $\bigoplus_{i=1}^{n} x_i = 0$ *but the vector* $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$ *does not, then*

$$Pr[\sigma_{xy} = 0] = \frac{1}{2}(1 + 2^{-n+1})$$

Proof. *A proof is similar to Lemma 1.*

6.2 Quadratic Approximation of $\omega_{(8)}$ with Linear Masking

Recall Equation (6). The bit $\omega_{(8)}$ can be expressed as a quadratic polynomial by using the previous $\omega_{(7)}$ bit recursively in a following way.

$$\begin{cases} \omega_{(0)} = s_{0,(0)} \oplus s_{16,(0)} \\ \omega_{(1)} = s_{0,(1)} \oplus s_{16,(1)} \oplus s_{0,(0)}s_{16,(0)} \\ \cdots \\ \omega_{(8)} = s_{0,(8)} \oplus s_{16,(8)} \oplus s_{0,(7)}s_{16,(7)} \oplus (s_{0,(7)} \oplus s_{16,(7)})(1 \oplus \omega_{(7)}) \end{cases} \tag{12}$$

If we apply the linear masking method, then,

$$Pr[\bigoplus_{t=\tau_1}^{\tau_6} \omega_{t,(8)} = 0] = Pr[\bigoplus_{t=\tau_1}^{\tau_6} (s_{0,(7)}s_{16,(7)} \oplus (s_{0,(7)} \oplus s_{16,(7)})(1 \oplus \omega_{(7)})) = 0] \tag{13}$$

Note that $\bigoplus_{t=\tau_1}^{\tau_6}(s_{0,(8)} \oplus s_{16,(8)}) = 0$.

Since the bit $\omega_{(7)}$ can be regarded as a (almost) balanced variable, the correlation of Equation (13) can be estimated by building a truth table where there are the condition that $\bigoplus_{t=\tau_1}^{\tau_6} s_{t,(7)} = \bigoplus_{t=\tau_1}^{\tau_6} s_{t+16,(7)} = 0$ but no condition on $\omega_{t,(7)}$, which corresponds the condition of Corollary 1. In result, a bit $\bigoplus_{t=\tau_1}^{\tau_6} \omega_{t,(8)}$ has the bias of around 2^{-5}. Experiments confirmed this result. See Appendix A.

6.3 Improved Bias of the Distinguisher

Recall again Equation (2) on NLF. If we replace $\alpha_{(8)}$ and $\beta_{(0)}$ by Approximations (4) and (5) respectively, but remain $\omega_{(8)}$, then, we build an approximation on NLF as follows.

$$\underbrace{s_{0,(25)} \oplus s_{0,(26)} \oplus s_{0,(28)} \oplus s_{0,(29)} \oplus s_{16,(26)} \oplus s_{16,(29)}}_{\alpha_{(8)}} \oplus \underbrace{s_{13,(29)} \oplus s_{13,(30)} \oplus z_{(29)} \oplus z_{(30)}}_{\beta_{(0)}}$$
$$\oplus \omega_{(8)} \oplus s_{1,(0)} \oplus s_{6,(0)} \oplus s_{13,(0)} \oplus K_{(0)} = z_{(0)} \tag{14}$$

with the bias of $2^{-4.1} \cdot 2^{-3.7} = 2^{-7.8}$.

Let us denote Approximation (14) simply as follows.

$$l_1(s) \oplus \omega_{(8)} = l_2(z) \tag{15}$$

where

$$\begin{aligned} l_1(s) &= s_{0,(25)} \oplus s_{0,(26)} \oplus s_{0,(28)} \oplus s_{0,(29)} \oplus s_{16,(26)} \oplus s_{16,(29)} \oplus s_{13,(29)} \\ &\quad \oplus s_{13,(30)} \oplus s_{1,(0)} \oplus s_{6,(0)} \oplus s_{13,(0)} \oplus K_{(0)} \\ l_2(z) &= z_{(0)} \oplus z_{(29)} \oplus z_{(30)} \end{aligned} \tag{16}$$

If we apply the linear masking method to Approximation (15),

$$\bigoplus_{t=\tau_1}^{\tau_6} (l_1(s) \oplus \omega_{(8)}) = \bigoplus_{t=\tau_1}^{\tau_6} (z_{(0)} \oplus z_{(29)} \oplus z_{(30)}) \tag{17}$$

Due to the linear connection of state bits by Equation (3) and Approximation (13), the left part of Approximation (17) vanishes with the probability of

$$\frac{1}{2}(1 + (2^{-7.8})^6 * 2^{-5}) = \frac{1}{2}(1 + 2^{-51.8}) \tag{18}$$

Therefore, in fact, a distinguisher of $\bigoplus_{t=\tau_1}^{\tau_6}(z_{(0)} \oplus z_{(29)} \oplus z_{(30)}) = 0$ has the bias of $2^{-51.8}$. Even though the distinguisher has not been changed, the usage of a quadratic terms improves the bias of distinguisher by a factor of 2, which reflects more accurate bias of the distinguisher.

7 Conclusions

In this paper, we show a distinguishing attack with linear masking against SOBER-128 stream cipher. This is the first work which presents an attack on SOBER-128. In particular, this work is interesting to eSTREAM project because the S-box of SOBER-128 is re-used for the NLS cipher [9, 5] which is one of the candidate stream ciphers. We estimate the correlation of a distinguisher by deriving a quadratic approximation on NLF.

Our attack shows that the correlation of the distinguisher with linear masking could be higher than the estimation at the paper [6] by considering a quadratic terms with a factor of at least 2.

References

1. The home page for eSTREAM. `http://www.ecrypt.eu.org/stream/`.
2. The home page for MUGI. `http://www.sdl.hitachi.co.jp/crypto/mugi/index-e.html`.
3. The home page for SNOW. `http://www.it.lth.se/cryptology/snow/`.
4. The home page for SOBER. `http://www.qualcomm.com.au/Sober.html`.
5. Joo Yeon Cho and Josef Pieprzyk. Linear distinguishing attack on NLS. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/018, 2006. `http://www.ecrypt.eu.org/stream/papersdir/2006/018.pdf`.
6. Don Coppersmith, Shai Halevi, and Charanjit Jutla. Cryptanalysis of stream ciphers with linear masking. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 in Lecture Notes in Computer Science, pages 515 – 532. Springer-Verlag, Jan. 2002.
7. P. Ekdahl and T.Johansson. Distinguishing attacks on SOBER-t16 and t32. In V. Rijmen J. Daemen, editor, *Fast Software Encryption*, volume 2365 in Lecture Notes in Computer Science, pages 210–224. Springer-Verlag, 2002.
8. P. Hawkes and G. Rose. Primitive specification for SOBER-128. http://www.qualcomm.com.au/Sober128.html, Apr. 2003.
9. P. Hawkes and G. Rose. Primitive specification for NLS. `http://www.ecrypt.eu.org/stream/nls.html` , Apr. 2005.
10. P.Hawkes and G.Rose. Sober. Primitive submitted to NESSIE by Qualcomm International, Sep. 2000.

A Experiments for Section 6.2

Experiments are begun by finding the initial states which would satisfy the following linear relation of the LFSR.

$$s_{t+\tau_1} \oplus s_{t+\tau_2} \oplus s_{t+\tau_3} \oplus s_{t+\tau_4} \oplus s_{t+\tau_5} \oplus s_{t+\tau_6} = 0$$

with $\tau_1 = 0, \tau_2 = 11, \tau_3 = 13, \tau_4 = 4 \cdot 2^{32} - 4, \tau_5 = 15 \cdot 2^{32} - 4, \tau_6 = 17 \cdot 2^{32} - 4$. Table 4 displays an example of initial states of $\tau_1, \ldots, \tau_6$. Note that all contents of the table are hexadecimal.

When $t = \tau_1$, we compute ω_8 by conducting $(s_0 \boxplus s_{16})_8$. (e.g. from the table, b0213cbe $\boxplus$ 7c0c7591 = 2c2db24f so that $\omega_8 = 0$) The same calculations are performed for $t = \tau_2$ to $t = \tau_6$. In result, we have 6 bits of ω_8 so that we can compute $\bigoplus_{t=\tau_1}^{\tau_6} \omega_{t,(8)}$. We carry on this process for $t = \{\tau_1 + 1, \ldots, \tau_6 + 1\}, t = \{\tau_1 + 2, \ldots, \tau_6 + 2\}$ and so on. New state is generated by the LFSR connection polynomial.

By counting the number of zeros (or ones) of the bit value $\bigoplus_{t=\tau_1}^{\tau_6} \omega_{t,(8)}$ at every clock, we can compute the probability which is the number of zeros (or ones) divided by the number of clocks.

The experiment shows that $\Pr[\bigoplus_{t=\tau_1}^{\tau_6} \omega_{t,(8)} = 0]$ is around $\frac{1}{2}(1 + 2^{-5})$ which was expected in Section 6.2.

Table 4. An example of initial states for the linear relation of LFSR

Register	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6
s_0	b0213cbe	81144c40	ea5f4936	80b626f2	7daca7b7	2670b88d
s_1	dee601f9	c0849eda	0da3e7a9	fd36421d	08f60296	e6013801
s_2	bb9d85af	18ce1254	a89d02b9	398e7a8a	80b626f2	b2f6c93a
s_3	6ddd2873	3937a5e3	19537890	e8eb08ef	fd36421d	5864bff2
s_4	3b3abd0f	6e162713	9e3a4268	6a8c43fa	398e7a8a	9814e104
s_5	98f4854a	e5ad513c	db7a3b35	387b5c1f	e8eb08ef	76b3bbb3
s_6	e77fc5c1	9983c08f	b100b099	0bfe370f	6a8c43fa	ae8ec122
s_7	b59aa80a	1e709998	a5c26138	1cfa270e	387b5c1f	2aa92bbb
s_8	9d0a4482	48ffd86a	b7368175	3b72bba8	0bfe370f	524f913a
s_9	2c927b9c	8c1aa656	a11f1bfb	983fe11e	1cfa270e	85520021
s_{10}	824e4c06	76126b97	713b00eb	79f12dc9	3b72bba8	c7e4b11b
s_{11}	d2389fa0	910a6bb8	a50ed952	03af6be3	983fe11e	7daca7b7
s_{12}	420962cd	0518c989	ec5437cf	da42b3d4	79f12dc9	08f60296
s_{13}	35949133	cbf0c10f	38fea16b	4583bc46	03af6be3	80b626f2
s_{14}	2be0a38b	3b5e5827	426bdfe3	75a1d586	da42b3d4	fd36421d
s_{15}	183186a9	83fe1b6a	db2c18b4	3cee43bb	4583bc46	398e7a8a
s_{16}	7c0c7591	d05172be	394a13c0	085dc986	75a1d586	e8eb08ef

Evaluating the Resistance of Stream Ciphers with Linear Feedback Against Fast Algebraic Attacks

An Braeken, Joseph Lano, and Bart Preneel

Katholieke Universiteit Leuven
Dept. Elect. Eng.-ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
{an.braeken, joseph.lano, bart.preneel}@esat.kuleuven.be

Abstract. In this paper we evaluate the resistance of stream ciphers with linear feedback against fast algebraic attacks. We summarize the current knowledge about fast algebraic attacks, develop new and more efficient algorithms to evaluate the resistance against fast algebraic attacks, study theoretical bounds on the attacks, and apply our methodology to the eSTREAM candidates SFINKS and WG as an illustration.

1 Introduction

Algebraic attacks (AAs) and fast algebraic attacks (FAAs), proposed in 2002 and 2003 respectively, are a very interesting and powerful development in the cryptanalysis of stream ciphers. So far, most papers on this subject (see e.g. [1, 5, 3, 6]) present ways to improve the algorithms to perform these attacks. Another interesting problem in this context is how to *evaluate* the resistance of stream ciphers against these attacks. Meier et al. [7] discuss how to evaluate the resistance of stream ciphers with linear feedback and a Boolean output function against AA. Therefore they define the property of *algebraic immunity* of a Boolean function.

In this paper, we will discuss how the resistance of such ciphers against *fast* algebraic attacks can be evaluated. We will approach this problem both from a theoretical and a practical perspective. As FAAs are very often the most efficient attacks against such designs, the evaluation of the resistance against FAAs is a very important tool when analyzing a stream cipher.

The outline of this paper is as follows. We first introduce some preliminaries in Sect. 2. In Sect. 3, we give an explicit description of the current state of the art in (fast) algebraic attacks. In Sect. 4, a new algorithm to evaluate the resistance of a Boolean function against a class of FAAs is presented. Section 5 studies some theoretical bounds on the resistance of Boolean functions against FAAs. We discuss the resistance of eSTREAM candidates SFINKS and WG against FAAs in Sect. 6 as an illustration and conclude in Sect. 7. Due to page restrictions, we could not include all results in this proceedings version. An extended version containing more results and a comparison with recent similar results will be made available.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 40–51, 2006.
© Springer-Verlag Berlin Heidelberg 2006

2 Preliminaries

The framework we develop applies to stream ciphers with a linear state update mechanism and a Boolean output function. The k-bit internal state is loaded with an initial state $\overline{s}_0 = (s_0, s_1, \dots s_{k-1})$, updated by a linear function L at each time t, i.e., $\overline{s}_t = L^t(\overline{s}_0)$ where $L^t = L \circ \cdots \circ L$. The Boolean output function f takes φ bits from the internal state and produces one key stream bit z_t.

We now introduce briefly some concepts on Boolean functions. A *Boolean function* f is a mapping from $\mathbb{F}_2^\varphi$ into $\mathbb{F}_2$. The *support* of f is defined as $\sup(f) = \{\overline{x} \in \mathbb{F}_2^\varphi : f(\overline{x}) = 1\}$. The cardinality of $\sup(f)$ represents the *weight* $\mathrm{wt}(f)$ of the function. A Boolean function f can be uniquely represented by means of its *algebraic normal form (ANF)*:

$$f(\overline{x}) = f(x_0, \dots, x_{\varphi-1}) = \bigoplus_{(a_0,\dots,a_{\varphi-1})\in\mathbb{F}_2^\varphi} c_f(a_0, \dots, a_{\varphi-1})\, x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}}, \quad \text{with } c_f(\overline{a}) = \bigoplus_{\overline{x}\preceq\overline{a}} f(\overline{x}), \tag{1}$$

where $\overline{x} \preceq \overline{a}$ means that $x_i \leq a_i$ for all $i \in \{0, \dots, n-1\}$. The *algebraic degree* of f, denoted by d_f, is defined as the highest number of variables in the terms $x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}}$ in the ANF of f.

Two functions f_1, f_2 on $\mathbb{F}_2^\varphi$ are said to be *affine equivalent* if

$$f_1(\overline{x}) = f_2(\overline{x}A \oplus \overline{b}) \oplus \overline{x} \cdot \overline{c} \oplus d, \tag{2}$$

where A is an invertible $\varphi \times \varphi$ matrix over $\mathbb{F}_2$, $\overline{b}, \overline{c} \in \mathbb{F}_2^\varphi$, $d \in \mathbb{F}_2$. Two functions are said to be *affine equivalent in the input variables* if and only if $\overline{c} = 0, d = 0$ in the above equation.

The lowest degree of a function g from $\mathbb{F}_2^\varphi$ into $\mathbb{F}_2$ for which $f(\overline{x}) \cdot g(\overline{x}) = 0$ for all $\overline{x} \in \mathbb{F}_2^\varphi$ or $(f(\overline{x}) \oplus 1) \cdot g(\overline{x}) = 0$ for all $\overline{x} \in \mathbb{F}_2^\varphi$ is called the *algebraic immunity (AI)* [7] of the function f. A function g is said to be an *annihilator* of f if $f(\overline{x}) \cdot g(\overline{x}) = 0$ for all $\overline{x} \in \mathbb{F}_2^\varphi$. It has been shown [5] that any function f with φ inputs has algebraic immunity at most $\lceil \frac{\varphi}{2} \rceil$.

We now introduce a similar notion that will be of relevance for FAAs. A Boolean function f on $\mathbb{F}_2^\varphi$ *satisfies a* (d_g, d_h)*-relation* (shortly $(d_g, d_h) - \mathcal{R}$) if a tuple of Boolean functions (g, h) of degree (d_g, d_h) exists such that $f(\overline{x}) \cdot g(\overline{x}) = h(\overline{x})$ for all $\overline{x} \in \mathbb{F}_2^\varphi$. This notion will only be relevant here when $d_g < d_h$.

In the rest of this paper, we denote the number of combinations with cardinality less than or equal to d in a set of u elements by $\mathcal{M}_d^u = \sum_{i=0}^{d} \binom{u}{i}$. Note that this number also represents the number of linearly independent functions (e.g. monomials) of degree less than or equal to d in $\mathbb{F}_2^u$.

3 Algebraic Attacks and Fast Algebraic Attacks

In this section we will describe AAs and FAAs, as developed in [5, 3] by Courtois and Meier. AAs are very basic and straightforward attacks. We write out the system of nonlinear equations between the initial state $\overline{s}_0$ and the key stream $(z_t)_{t\geq 0}$ and try to solve this. These attacks turn out to be very powerful. The

system of equations we are trying to solve consists of equations $f(L^i(\overline{s}_0)) = z_i$. Clearly, this original system of equations has k unknowns (corresponding with the k bits of the initial state) and degree d_f.

3.1 Algebraic Attacks

In a standard AA we will try to reduce the degree of the above system of equations before we solve it. This will lower the attack complexity considerably.

First, we need to perform a *relation search step*. Here we search for nonzero functions g and/or h of lower degree d_g and/or d_h both smaller than d_f, such that the following holds for all $\overline{x} \in \mathbb{F}_2^{\varphi}$:

$$\begin{cases} f(\overline{x}) \cdot g(\overline{x}) = 0 \\ (f(\overline{x}) \oplus 1) \cdot h(\overline{x}) = 0\,. \end{cases} \tag{3}$$

It is easy to see that we can then replace each equation in by a lower degree equation as follows:

$$\begin{array}{l} g(L^i(\overline{s}_0)) = 0, \forall\ i \text{ where } z_i = 1 \\ h(L^i(\overline{s}_0)) = 0, \forall\ i \text{ where } z_i = 0\,. \end{array} \tag{4}$$

This new system of equations can be solved much faster because of its lower degree. To facilitate the complexity analysis, we use linearization (replacing each monomial by a new variable) followed by Gaussian elimination of this overdetermined system of equations. Consequently, the time complexity of solving the system of equations (4) is of order $(\mathcal{M}_{d_g}^k)^\omega$, with $\omega = \log_2(7) \approx 2.807$ (Strassen's exponent [9]). The data complexity is about $\mathcal{M}_{d_g}^k$ but can be lowered if multiple functions g and/or h exist.

The complexity of the AA clearly decreases rapidly with the degree of the multiples of f and $f \oplus 1$. Hence the efficiency of an AA depends on the lowest degree of any multiple. This corresponds to the algebraic immunity of the Boolean function, as defined above. So in essence to determine the resistance of a design against the AA, all that needs to be done is to make sure that the AI of its output function is sufficiently high. We will explain how this can be computed in Sect. 4.

3.2 Fast Algebraic Attacks

FAAs can be much more efficient than the standard AAs. In the FAA introduced by Courtois [3], the attacker tries to decrease the degree d of the system of equations even further by searching for relations between the initial state and several bits of the output function simultaneously, *i.e.*, equations of the form

$$F(\overline{s}_0, z_t, z_{t+1}, \ldots, z_{t+T}) = 0, \forall t, \tag{5}$$

with $F(\overline{s}_0, \overline{z}) = g(\overline{s}_0, \overline{z}) \oplus h(\overline{s}_0)$ and $d_g < d_h$. See [6] for a general description. Let us now restrict to the case $T = 1$ in (5) and thus obtain the equations

$$z_i g(L^i(\overline{k})) \oplus h(L^i(\overline{k})) = 0, i \geq 0\,, \tag{6}$$

where (g, h) represents the tuple of functions with low degrees (d_g, d_h) and $d_g < d_h$ for which $f \cdot g = h$.

So far, FAAs mounted on actual stream ciphers with linear update mechanism all use equations of type (6). These are the FAAs we are going to consider here. It is an open problem whether better attacks of type (5) can be found for actual stream ciphers.

The considered attack can be described by the following simple framework, as briefly described by Courtois in [3] and more explicitly in [4].

1. *Relation search step* : We need to find $(d_g, d_h) - \mathcal{R}$ for our Boolean function. As defined above, f satisfies $(d_g, d_h) - \mathcal{R}$ if an equation exists of of the following form, where g and h are nonzero and $d_g < d_h$:
$$f(\overline{x}) \cdot g(\overline{x}) = h(\overline{x}) \qquad \forall \overline{x} \in \mathbb{F}_2^{\varphi}\,. \tag{7}$$
Multiplying the original equations by g, our system of equations now becomes of degree d_h:
$$\begin{cases} h(\overline{s}_0) & = z_0 \cdot g(\overline{s}_0) \\ h(L(\overline{s}_0)) & = z_1 \cdot g(L(\overline{s}_0)) \\ h(L^2(\overline{s}_0)) & = z_2 \cdot g(L^2(\overline{s}_0)) \\ \dots \end{cases} \tag{8}$$
2. *Precomputation step* : We now reduce the degree of (8) further from d_h to d_g in a precomputation step. Because the combination of linear recurring sequences is again a linear recurring sequence, there *always* exists a single linear combination of every $\mathcal{M}_{d_h}^k + 1$ consecutive bits such that for every $t \geq 0$:
$$\bigoplus_{i=0}^{\mathcal{M}_{d_h}^k} \alpha_i \cdot h(L^{i+t}(\overline{s}_0)) = 0, \tag{9}$$
and therefore our system of equations (8) becomes of degree d_g and has the following form:
$$\begin{cases} \bigoplus_{i=0}^{\mathcal{M}_{d_h}^k} \alpha_i \cdot z_i \cdot g(L^i(\overline{s}_0)) & = 0 \\ \bigoplus_{i=0}^{\mathcal{M}_{d_h}^k} \alpha_i \cdot z_{i+1} \cdot g(L^{i+1}(\overline{s}_0)) & = 0 \\ \bigoplus_{i=0}^{\mathcal{M}_{d_h}^k} \alpha_i \cdot z_{i+2} \cdot g(L^{i+2}(\overline{s}_0)) & = 0 \\ \dots \end{cases} \tag{10}$$
The complexity of this precomputation step has been shown to be $\mathcal{M}_{d_h}^k \cdot (\log_2 \mathcal{M}_{d_h}^k)^2$ (see Hawkes and Rose, [6]).
3. *Substitution step* : As we obtain an actual key stream and use it in a FAA, we need to substitute this key stream into (10). The best algorithm to do this so far has been described in [6] and has complexity $2 \cdot \mathcal{M}_{d_g}^k \cdot \mathcal{M}_{d_h}^k \cdot \log_2 \mathcal{M}_{d_h}^k$.
4. *Solving step* : We solve the system (10) by linearization. The time complexity to do so is of order $(\mathcal{M}_{d_g}^k)^{\omega}$. The data needed is $\mathcal{M}_{d_g}^k + \mathcal{M}_{d_h}^k \approx \mathcal{M}_{d_h}^k$. This data complexity can be reduced by a factor if we find several linear combinations, but then the substitution complexity will increase by the same factor.

The following observation is very important : to evaluate the resistance of a stream cipher with linear feedback and Boolean output function against this particular FAA, all we need to do is to perform step 1 (the relation search) for its Boolean function. Namely, we need to find out which $(d_g, d_h) - \mathcal{R}$ the Boolean function satisfies. In Sect. 4, we will show a new algorithm which performs this step much faster than previously thought possible, and which hence makes it possible to evaluate the resistance of stream cipher with linear feedback using relatively large Boolean functions against such a FAA (see Sect. 6 for some examples).

4 Algorithms for Computing AI and $(d_g, d_h) - \mathcal{R}$

As we explained in Sect. 3, to evaluate the resistance of a generator against the considered classes of (fast) AAs, all that needs to be done is performing the corresponding relation search step for its Boolean function. We now describe the algorithms to do so. In Sect. 4.1 we describe the algorithms from [7] to find relations for mounting AAs, but explain them in a more elegant way. This will help to understand the new algorithm we will develop in Sect. 4.2, and why this algorithm is much more efficient than previously thought possible.

4.1 Computing the AI of a Boolean Function

The aim is to find or to disprove the existence of a function $g(\overline{x})$ of degree d_g such that $f(\overline{x}) \cdot g(\overline{x}) = 0$ for all inputs $\overline{x}$. The unknowns we need to find are the coefficients $c_g(\overline{a})$ with $wt(\overline{a}) \leq d_g$ in the ANF of g:

$$g(\overline{x}) = \bigoplus_{\substack{\overline{a} \in \mathbb{F}_2^{\varphi} \\ wt(\overline{a}) \leq d_g}} c_g(\overline{a})\, x_0^{a_0} \ldots x_{\varphi-1}^{a_{\varphi-1}}\,. \tag{11}$$

Hence the number of unknown coefficients is $\mathcal{M}_{d_g}^{\varphi}$. The equations we can use to find these unknowns are found as follows: the requirement $f(\overline{x}) \cdot g(\overline{x}) = 0$ can be easily seen to correspond to the requirement that $g(\overline{x}) = 0$ whenever $f(\overline{x}) = 1$. As a good cryptographic Boolean function should be balanced, we have exactly $2^{\varphi-1}$ such equations. As we are looking for $d_g < \lceil \frac{\varphi}{2} \rceil$, the number of equations $2^{\varphi-1}$ is always larger than the number of unknowns $\mathcal{M}_{d_g}^{\varphi}$.

Solving the above system of equations corresponds to algorithm 1 in [7]. The complexity of solving this system naively is of order $2^{\varphi-1} \cdot (\mathcal{M}_{d_g}^{\varphi})^2$. However, we can also just solve a subsystem of $\mathcal{M}_{d_g}^{\varphi}$ equations and check whether its solution fits the other equations. Then we get a complexity in the order of $(\mathcal{M}_{d_g}^{\varphi})^3$.

Algorithm 2 of [7] improves this complexity. In this algorithm, a specific choice is made for the subsystem of $\mathcal{M}_{d_g}^{\varphi}$ equations that we will solve. Denote the system we are solving by

$$M \cdot \overline{c}_g = \overline{0}\,, \tag{12}$$

where $\overline{c}_g$ is the $\mathcal{M}^{\varphi}_{d_g} \times 1$ column matrix containing the ANF coefficients $c_g(\overline{a})$ ordered lexicographically on $\overline{a}$, and M is a $\mathcal{M}^{\varphi}_{d_g} \times \mathcal{M}^{\varphi}_{d_g}$ matrix of which we choose the rows as follows: the first row corresponds with the equation $g(\overline{x}) = 0$ with $wt(\overline{x}) = 0$ (if it exists, i.e., if $f(\overline{0}) = 1$). The following rows correspond to the equations (if they exist) where the Hamming weight of the input vector is 1, 2, ..., until Hamming weight d_g, all ordered lexicographically. By now we have about $\frac{1}{2}\mathcal{M}^{\varphi}_{d_g}$ rows of our matrix M. We fill up the remaining rows of M by choosing other values $\overline{x}$ at random.

Now we look at the upper part of this matrix, corresponding to the rows up to Hamming weight d_g. It is easy to see that this part of the matrix is in *echelon form* and its size is of order $\frac{1}{2}\mathcal{M}^{\varphi}_{d_g} \times \mathcal{M}^{\varphi}_{d_g}$. From the structure of the problem presented, it is easy to see that this echelon form has order of $\frac{1}{2}\mathcal{M}^{\varphi}_{d_g}$ pivot columns, $\frac{1}{2}\binom{\varphi}{d_g}$ zero columns (which we can ignore) and $\frac{1}{2}\mathcal{M}^{\varphi}_{d_g-1}$ nonzero nonpivot columns. The solution of this underdetermined system of equations can be found efficiently, with a complexity of order $\frac{1}{4}\mathcal{M}^{\varphi}_{d_g} \cdot (\mathcal{M}^{\varphi}_{d_g} + \mathcal{M}^{\varphi}_{d_g-1})$ operations.

We now substitute these solutions into the remaining $\frac{1}{2}\mathcal{M}^{\varphi}_{d_g}$ equations. Only the nonzero pivot columns need to be substituted into the system, resulting in a substitution complexity of order $\frac{1}{8}(\mathcal{M}^{\varphi}_{d_g})^2 \cdot \mathcal{M}^{\varphi}_{d_g-1}$, and we obtain a system of $\frac{1}{2}\mathcal{M}^{\varphi}_{d_g}$ equations in $\frac{1}{2}\mathcal{M}^{\varphi}_{d_g}$ unknowns, which can be solved by Gaussian elimination in order of $\frac{1}{8}(\mathcal{M}^{\varphi}_{d_g})^3$ operations.

This approach corresponds to algorithm 2 in [7]. The last step will be the dominating step in terms of time complexity. To make the algorithm memory efficient we will process the matrix row by row and only store the nonzero nonpivot elements for later use, resulting in a memory complexity of order $\frac{1}{4}\mathcal{M}^{\varphi}_{d_g}\mathcal{M}^{\varphi}_{d_g-1}$.

4.2 Computing the $(d_g, d_h) - \mathcal{R}$ for a Boolean Function

The gain of algorithm 2 compared to algorithm 1 for computing the AI is only modest because the final system we need to solve still has half the number of unknowns as the original system. We will now show that for the search of $(d_g, d_h) - \mathcal{R}$, we can do much better.

Our aim is to find out whether the Boolean function satisfies $(d_g, d_h) - \mathcal{R}$ for certain values $d_g < d_h$ at which the complexity of a FAA would be less than exhaustive search. The unknowns we need to find are the coefficients $c_g(\overline{a})$ with $wt(\overline{a}) \le d_g$ and $c_h(\overline{a})$ with $wt(\overline{a}) \le d_h$ in the ANFs of g and h:

$$\begin{aligned} g(\overline{x}) &= \bigoplus_{\substack{\overline{a}\in\mathbb{F}_2^{\varphi} \\ wt(\overline{a})\le d_g}} c_g(\overline{a})\, x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}} \\ h(\overline{x}) &= \bigoplus_{\substack{\overline{a}\in\mathbb{F}_2^{\varphi} \\ wt(\overline{a})\le d_h}} c_h(\overline{a})\, x_0^{a_0} \dots x_{\varphi-1}^{a_{\varphi-1}} . \end{aligned} \tag{13}$$

Hence the number of unknown coefficients is $\mathcal{M}^{\varphi}_{d_h} + \mathcal{M}^{\varphi}_{d_g}$.

Previously, it was thought that the complexity of solving this system of equations is of order $(\mathcal{M}^{\varphi}_{d_h})^3$, making it hard to evaluate the resistance against FAAs

even for functions with modest input dimension. We will now show a new algorithm that can solve the problem in time complexity of order $\mathcal{M}_{d_h}^{\varphi} \cdot (\mathcal{M}_{d_g}^{\varphi})^2 + (\mathcal{M}_{d_h}^{\varphi})^2$. This makes evaluation of the resistance against FAAs achievable for larger Boolean functions, as shown by the examples in Sect. 6.

The equations used to find the unknown coefficients of the ANFs of g and h are as follows: the requirement $f(\overline{x}) \cdot g(\overline{x}) = h(\overline{x})$ for all $\overline{x} \in \mathbb{F}_2^{\varphi}$ gives us a condition on the ANF coefficients of g and h for *every* value of $\overline{x}$. This results in an equation that can be of one of the following two types:

$$\begin{array}{ll} g(\overline{x}) = h(\overline{x}) & \text{whenever } f(\overline{x}) = 1 \\ h(\overline{x}) = 0 & \text{whenever } f(\overline{x}) = 0\,. \end{array} \tag{14}$$

The number of equations obtained, 2^{φ}, will always be larger than the number of unknowns $\mathcal{M}_{d_h}^{\varphi} + \mathcal{M}_{d_g}^{\varphi}$ in situations of practical interest.

The naive solution is again to solve this system of equations, which has complexity of order $2^{\varphi} \cdot (\mathcal{M}_{d_h}^{\varphi} + \mathcal{M}_{d_g}^{\varphi})^2$. However, we can also just solve a subsystem of $\mathcal{M}_{d_h}^{\varphi} + \mathcal{M}_{d_g}^{\varphi}$ equations and see if its solution fits the other equations. Then we get a complexity in the order of $(\mathcal{M}_{d_h}^{\varphi} + \mathcal{M}_{d_g}^{\varphi})^3$.

It is possible to significantly reduce the complexity further by a good choice of the subsystem to be solved and by making use of its structure. Denote again the system we are solving by

$$M \cdot \overline{c}_{g,h} = \overline{0}\,, \tag{15}$$

where $\overline{c}_{g,h}$ is the $(\mathcal{M}_{d_h}^{\varphi} + \mathcal{M}_{d_g}^{\varphi}) \times 1$ column matrix containing the ANF coefficients $c_h(\overline{a})$ ordered lexicographically, where we intercalate the coefficients $c_g(\overline{a})$ right before the corresponding $c_h(\overline{a})$ as long as such coefficients exist - recall that $d_g < d_h$. The matrix M is a $(\mathcal{M}_{d_h}^{\varphi} + \mathcal{M}_{d_g}^{\varphi}) \times (\mathcal{M}_{d_h}^{\varphi} + \mathcal{M}_{d_g}^{\varphi})$ matrix of which we choose the rows as follows: the first row corresponds to the equation (14) with $wt(\overline{x}) = 0$. The following φ rows correspond to the equations (14) with $wt(\overline{x}) = 1$ ordered lexicographically, followed by the $\binom{\varphi}{2}$ rows corresponding to the equations (14) with $wt(\overline{x}) = 2$. We continue this procedure until the $\binom{\varphi}{d_h}$ rows corresponding to the equations (14) with $wt(\overline{x}) = d_h$. The remaining $\mathcal{M}_{d_g}^{\varphi}$ rows of M are then filled up by choosing other values $\overline{x}$ at random for obtaining additional equations (14).

Again, we can reduce the complexity significantly by looking at the structure of this M. Let's look at the $\mathcal{M}_{d_h}^{\varphi}$ first rows of this matrix. It is easy to see that this part of the matrix is in *echelon form* and its size is exactly $\mathcal{M}_{d_h}^{\varphi} \times (\mathcal{M}_{d_h}^{\varphi} + \mathcal{M}_{d_g}^{\varphi})$. The solution of this underdetermined system of equations can be found efficiently, requiring order of $(\mathcal{M}_{d_h}^{\varphi})^2 + \mathcal{M}_{d_h}^{\varphi}\mathcal{M}_{d_g}^{\varphi} \approx (\mathcal{M}_{d_h}^{\varphi})^2$ operations. The $\mathcal{M}_{d_h}^{\varphi}$ pivot columns correspond to the coefficients $c_h(\overline{a})$ with $wt(\overline{a}) \leq d_h$, the $\mathcal{M}_{d_g}^{\varphi}$ nonpivot columns correspond to the coefficients $c_g(\overline{a})$ with $wt(\overline{a}) \leq d_g$.

We now substitute this into the remaining equations. We need to substitute order of $\mathcal{M}_{d_h}^{\varphi}$ coefficients by $\mathcal{M}_{d_g}^{\varphi}$ coefficients and do this in $\mathcal{M}_{d_g}^{\varphi}$ rows, resulting in a substitution complexity of order $\mathcal{M}_{d_h}^{\varphi} \cdot (\mathcal{M}_{d_g}^{\varphi})^2$. We can then solve this

system to obtain the coefficients $c_g(\overline{a})$ through Gaussian elimination, requiring order of $(\mathcal{M}^{\varphi}_{d_g})^3$ operations. Finally, we introduce the solution for the coefficients $c_g(\overline{a})$ with $wt(\overline{a}) \leq d_g$ into the upper part of the matrix and can efficiently find the solution for the coefficients $c_h(\overline{a})$ with $wt(\overline{a}) \leq d_h$.

The gain in complexity is much more important here than in the case of the previous section. Adding up the complexities of the steps and discarding terms that will not be relevant for the analysis, the time complexity will be dominated by either finding the solution of the echelon matrix or by the substitution complexity. Our algorithm hence has a time complexity of order $\mathcal{M}^{\varphi}_{d_h} \cdot (\mathcal{M}^{\varphi}_{d_g})^2 + (\mathcal{M}^{\varphi}_{d_h})^2$. To make the algorithm memory efficient, we will process the matrix row by row and only store the nonpivot elements for later use, resulting in a memory complexity of order $\mathcal{M}^{\varphi}_{d_h}\mathcal{M}^{\varphi}_{d_g}$.

5 Theoretical Bounds on the Existence of $(d_g, d_h) - \mathcal{R}$

Relation Between φ and $(d_g, d_h) - \mathcal{R}$. We will first look into the relation between the existence of $(d_g, d_h) - \mathcal{R}$ and the number of inputs φ of the Boolean function. To do so, we first investigate the affine invariance of $(d_g, d_h) - \mathcal{R}$. For AI, it holds that the AI of affine equivalent functions can differ with one. Only for functions that are affine equivalent in the input variables, the AI is invariant. We now show that $(d_g, d_h) - \mathcal{R}$ satisfies a stronger invariance:

Theorem 1. *The $(d_g, d_h) - \mathcal{R}$ (with $d_g < d_h$) of Boolean functions is an affine invariant property.*

Proof. Let $f(\overline{x})$ satisfy a $(d_g, d_h) - \mathcal{R}$ with $d_g < d_h$:

$$f(\overline{x}) \cdot g(\overline{x}) = h(\overline{x}). \tag{16}$$

Then for any function f' that is affine equivalent to f expressed by Equation (2), the following relation holds:

$$\begin{aligned} f'(\overline{x}) \cdot g(\overline{x}A \oplus \overline{b}) &= (f(\overline{x}A \oplus \overline{b}) \oplus \overline{x} \cdot \overline{c} \oplus d) \cdot g(\overline{x}A \oplus \overline{b}) \\ &= [f(\overline{x}A \oplus \overline{b}) \cdot g(\overline{x}A \oplus \overline{b})] \oplus [(\overline{x} \cdot \overline{c} \oplus d) g(\overline{x}A \oplus \overline{b})] \end{aligned} \tag{17}$$

In the right hand side, the first term is equal to $h(\overline{x}A \oplus \overline{b})$ because of (16), and the second term has degree $\leq d_h$ because $d_g < d_h$. So (17) shows that f' satisfies $(d_g, d_h) - \mathcal{R}$. □

A bound on the relation between $(d_g, d_h) - \mathcal{R}$ and φ has already been given in [3, Theorem 7.2.1] by Courtois:

Theorem 2. *A Boolean function f always satisfies $(d_g, d_h) - \mathcal{R}$ for any d_g and d_h such that $d_g + d_h \geq \varphi$.*

Relation Between d_f and $(d_g, d_h) - \mathcal{R}$. Second, we investigate the bounds posed by the degree of the Boolean function on the existence of $(d_g, d_h) - \mathcal{R}$. The following straightforward relation exists between $(d_g, d_h) - \mathcal{R}$ and the degree (also mentioned in [3, Theorem 7.1.1]) .

Theorem 3. *If f has degree d_f, then f satisfies $(i, d_f + i) - \mathcal{R}$ for every $i < d_f$.*

Relation Between AI and $(d_g, d_h) - \mathcal{R}$. We now show that if f satisfies $(d_g, d_h) - \mathcal{R}$, the parameter d_h is greater than or equal to $\mathrm{AI}(f)$, due to the following relation with the annihilators of f and $f \oplus 1$.

Lemma 1. *Let f, g, h be three Boolean functions on $\mathbb{F}_2^\varphi$. The equation $f \cdot g = h$ is satisfied if and only if both $f \cdot (g \oplus h) = 0$ and $(f \oplus 1) \cdot h = 0$ hold simultaneously. In other words, if and only if $g \oplus h$ is an annihilator of f and h is an annihilator of $f \oplus 1$.*

Proof. Let $f \cdot g = h$. Multiplying both sides of the equation with f leads to $f^2 \cdot g = f \cdot h$. Consequently, $f \cdot g = f \cdot h$ and $f \cdot h = h$. The other side of the implication is trivial. □

As a consequence, for every d_h greater than or equal to $\mathrm{AI}(f)$, the lowest possible d_g such that f satisfies $(d_g, d_h) - \mathcal{R}$ could be found by searching the lowest degree function which is a linear combination of functions belonging to the set of annihilators with degree d_h of f and $f \oplus 1$. We have implemented an algorithm that uses this methodology. This algorithm is less efficient than the one described in Sect. 4.2, but the implementation effort of this new algorithm is more complex and is work in progress.

6 Application to Concrete Designs

6.1 SFINKS

SFINKS [2] is a filter generator with 80-bit secret key and 256-bit internal state. The filter function has 17 inputs and degree 15 ($\varphi = 17$ and $d_f = 15$). The resistance against AAs was measured using the algorithm of [7], resulting in an AI of 6. To analyze the resistance against FAAs, we need to check whether the function satisfies $(d_g, d_h) - \mathcal{R}$ in all cases where the corresponding attack complexity is smaller than 2^{80}. Table 1 presents the total complexity order (i.e., the precomputation, substitution and solving complexities together) of the FAA for an arbitrary filter generator with internal size 256 and filter function which satisfies $(d_g, d_h) - \mathcal{R}$, only when $d_h \geq 6$ because of Lemma 1.

Theoretical bounds. From Theorem 2, it follows that f satisfies $(1, 16) - \mathcal{R}$, $(2, 15) - \mathcal{R}$, $(3, 14) - \mathcal{R}$, ... but none of these are a threat. From Theorem 3, it follows that f satisfies $(1, 16) - \mathcal{R}$, $(2, 17) - \mathcal{R}$,... which is not a problem as well.

Table 1. $\log_2$ of order of complexities of FAA for a 256-bit state if f satisfies $(d_g, d_h) - \mathcal{R}$

$d_g \backslash d_h$	6	7	8	9	10	11	12
1	**52.73**	**58.10**	**63.20**	**68.12**	**72.88**	**77.47**	81.93
2	**59.73**	**65.10**	**70.20**	**75.12**	**79.88**	84.47	88.93
3	**66.14**	**71.48**	**76.61**	81.53	86.28	90.88	95.34
4	**76.94**	**77.47**	82.60	87.52	92.27	96.87	101.33
5	92.83	92.83	92.83	93.18	97.94	102.53	106.99

Table 2. $\log_2$ of order of complexities of FAA for a 319-bit state if f satisfies $(d_g, d_h)-\mathcal{R}$

$d_g \backslash d_h$	1	2	3	4	5	6	7	8	9	10	11
1	**23.36**	**28.93**	**36.17**	**42.84**	**49.09**	**55.03**	**60.70**	**66.15**	**71.40**	**76.48**	81.41
2	0	**43.91**	**43.91**	**50.16**	**56.41**	**62.35**	**68.02**	**73.47**	**78.72**	83.80	88.73
3	0	0	**62.79**	**62.79**	**63.14**	**69.07**	**74.75**	80.19	85.45	90.53	95.45
4	0	0	0	80.50	80.50	80.50	81.05	86.50	91.76	96.84	101.76

Searching for relations. It is possible that we can improve on the theoretical bounds by actually searching for relations. With our algorithm of Sect. 4.2, this can be done easily. We only need to verify the existence of relations for which the attack complexity is less than 2^{80}. The complexity of doing so can be calculated to be of order 2^{39}. Courtois [4] already checked for the existence of such relations, using the old algorithm but which found the relations in a few days. We repeated his experiments with the old algorithm and confirm his simulations.

In particular, Courtois found that f satisfies $(4,6)-\mathcal{R}$, $(3,7)-\mathcal{R}$ and $(2,8)-\mathcal{R}$. Note that if one assumes that the number of key stream bits is limited to 2^{40} (as stated in [2]), not all of these attacks are possible. But still the $(4,6)-\mathcal{R}$ and $(3,7)-\mathcal{R}$ (using several relations simultaneously for the latter) attacks are practical and can be mounted faster than exhaustive key search.

Remark 1. Due to the particular form of the filter function f in SFINKS, i.e., $f(\overline{x}, x_{16}) : \mathbb{F}_2^{17} \to \mathbb{F}_2 : \overline{x} \mapsto f_1(\overline{x}) \oplus x_{16}$, where f_1 is a Boolean function on $\mathbb{F}_2^{16}$ affine equivalent to the trace function of the inverse function on $\mathbb{F}_2^{16}$, we can restrict us to the computation of the $(d_g, d_h)-\mathcal{R}$ with $d_g < d_h$ of f_1. This follows from the observation that

$$f_1 \cdot g = h \Rightarrow (f_1 \oplus x_{16}) \cdot g = h \oplus x_{16} \cdot g$$
$$f_1 \cdot g = h \oplus x_{16} \cdot g \Leftarrow (f_1 \oplus x_{16}) \cdot g = h\,.$$

In general, we have the following theorem.

Theorem 4. *Let f_i be a Boolean function on $\mathbb{F}_2^i$ that satisfies $(e_i, d_i) - \mathcal{R}$ for $i = 1, 2$. Then $f_1 \oplus f_2$ on $\mathbb{F}_2^{n_1+n_2}$ satisfies also $(e_1, \max\{\deg(f_2) + e_1, d_1\}) - \mathcal{R}$ and $(e_2, \max\{\deg(f_1) + e_2, d_2\}) - \mathcal{R}$.*

6.2 WG

The stream cipher WG [8] is also a simple filter generator with 80-bit secret key, 319-bit internal state and a filter function with 29 inputs and degree 11. The AI could not be measured yet. For the FAA, we should again check whether f satisfies $(d_g, d_h) - \mathcal{R}$ for all cases where the attack complexity is less than 2^{80}. Table 2 presents these complexity orders.

Theoretical Bounds. From Theorem 2, it follows that f satisfies $(1, 28) - \mathcal{R}$, $(2, 27) - \mathcal{R}$, $(3, 26) - \mathcal{R}$, ... but none of these are a threat. From Theorem 3, it follows that f satisfies $(1, 12) - \mathcal{R}$, $(2, 13) - \mathcal{R}$,...

Table 3. $\log_2$ of order of complexities for the Relation Search Step for WG with new algorithm

$d_g \backslash d_h$	1	2	3	4	5	6	7	8	9	10	11
1	14.77	19.15	24.28	29.58	34.33	38.49	42.12	45.25	47.95	50.24	0
2	0	26.31	29.57	32.50	35.52	38.88	42.24	45.30	47.97	0	0
3	0	0	36.00	38.76	41.17	43.29	45.23	0	0	0	0

Let us investigate this $(1, 12)-\mathcal{R}$ into more detail. We know that there exist at most 2^{30} equations of this type. These equations are simply found by multiplying the function with an arbitrary affine function which increases the degree with at most one. The required key stream length is equal to $2^{70.73}$ and the attack complexities are as follows.

- Precomputation complexity: $2^{81.958}$
- Substitution complexity: $2^{86.19}$
- Solving complexity: $2^{23.36}$

In the proposal of the stream cipher WG [8], the key stream has been restricted to sequences of length 2^{45}. A FAA may still exist if at least $2^{25.73}$ out of the 2^{30} equations are linearly independent. In this situation, the substitution complexity will increase with a factor of $2^{25.73}$, leading to a complexity of $2^{111.4}$.

Searching for relations. It may also be that better FAAs can be performed on WG. To check this, we need to run our algorithm and search for relations. Table 3 presents the complexity of our algorithm for determining if f is $(d_g, d_h)-\mathcal{R}$. We expect that most of these values are attainable when implementing the algorithm on a modern PC. On the contrary, the old algorithm would have an infeasible complexity of order 2^{75}.

7 Conclusion

In this paper, we have presented a framework to evaluate the resistance of memoryless nonlinear filters and combiners against FAAs, using the (non-)existence of $(d_g, d_h)-$relations to assess the security. We developed a new algorithm to efficiently calculate these relations, and presented some bounds that relate the resistance against FAAs to some properties of the Boolean function. Finally, we have illustrated the application of our approach on two eSTREAM candidates, SFINKS (as previously done by Courtois) and WG. Further study of FAAs is still required, especially towards general classes of these attacks.

Acknowledgements. This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme ECRYPT. An Braeken is an FWO Research Assistant, sponsored by the Fund for Scientific Research - Flanders (Belgium), Joseph Lano is financed by a PhD grant of the Institute

for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). The authors also would like to thank Nicolas Courtois and the anonymous referees for their useful comments.

References

1. F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In D. Boneh, editor, *Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 162–175. Springer-Verlag, 2003.
2. A. Braeken, J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede. SFINKS: A synchronous stream cipher for restricted hardware environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/035, 2005. `http://www.ecrypt.eu.org/stream`.
3. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In D. Boneh, editor, *Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer-Verlag, 2003.
4. N. Courtois. Cryptanalysis of SFINKS. In M. Rhee and B. Lee, editors, *Information Security and Cryptology - ICISC 2005*, number 3935 in Lecture Notes in Computer Science. Springer-Verlag, 2005.
5. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.
6. P. Hawkes and G. Rose. Rewriting variables: The complexity of fast algebraic attacks on stream ciphers. In M. Franklin, editor, *Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 390–406. Springer-Verlag, 2004.
7. W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In C. Cachin and J. Camenisch, editors, *Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer-Verlag, 2004.
8. Y. Nawaz and G. Gong. The WG stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/033, 2005. `http://www.ecrypt.eu.org/stream`.
9. V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

Ensuring Fast Implementations of Symmetric Ciphers on the Intel Pentium 4 and Beyond*

Matt Henricksen and Ed Dawson

Information Security Institute, Queensland University of Technology,
GPO Box 2434, Brisbane, Queensland, 4001, Australia
{m.henricksen, e.dawson}@qut.edu.au

Abstract. Cipher design is a multi-faceted process. Many designers focus on security, or present novel designs, but neglect to consider the impact on their ciphers' efficiency. This paper presents simple guidelines for ensuring efficient symmetric cipher implementations on the Intel Pentium 4 and associated architectures. The paper examines the suitability of a handful of ECRYPT eSTREAM ciphers for the platform, including Dragon, HC-256, MAG, Mir-1, Phelix, and Py.

Keywords: Stream cipher, Implementation, Intel Pentium 4, Dragon, HC-256, MAG, Mir-1, Py, RC4.

1 Introduction

The days of slow symmetric ciphers are over. It is widely considered that efficiency of ciphers is nearly as important as their security. The benchmark set in software by the Advanced Encryption Standard (AES) [4] is around fifteen cycles per byte on the Intel Pentium 4 [13]. Symmetric ciphers that do not equal or surpass this benchmark are widely regarded as non-competitive [1]. Word-based stream ciphers are, in many cases ([1], [3], [18]), notably faster than block ciphers, while not demonstrably less secure.

In 2005, the ECRYPT NoE eSTREAM stream cipher project released a call [5] for stream cipher primitives. Of the thirty-five candidates, twenty-two were profiled as suitable for software implementation, with efficiencies ranging from three cycles [1] through to a sluggish 1,400 cycles per byte [11]. The diversity of results reveals that many cipher designers do not understand either the need for efficiency, or how to achieve it in their ciphers.

This paper discusses the design of ciphers with respect to the Intel Pentium 4. As this architecture is backwards compatible with the entire Intel x86 line, most of the advice holds for the other members of the Pentium family, such as the Pentium III. It also holds for the Pentium-D, which contains two Pentium 4

* This work was partially funded by Australian Research Council Discovery Project Grant DP0450920.

[1] For example, in the second round of the ECRYPT eSTREAM project, stream ciphers that did not compete with the block cipher AES were "archived" [6].

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 52–63, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Prescott chips in a single package. Much of the advice is, at a high level, similar to that found in the paper of Schneier and Whiting [15], published nearly a decade earlier. This is a natural consequence of Intel's strategy of backwards compatibility in the Pentium line of processors.

The guidelines within this paper are prioritized as *High*, *Medium* or *Low* according to their impact. They are presented at an algorithmic rather than implementation level. It is easy to obtain an efficient implementation of a cipher that has been well designed. Conversely, it may prove to be impossible to derive a fast implementation if attention is not paid to efficiency during the cipher's design phase. The designer of a cipher and its optimizer are frequently different people. This paper is presented to help the designer to understand how to facilitate the job of the optimizer. For a good discussion about efficient implementation of ciphers on recent architectures, see [13].

Section 2 describes how to commence the task of cipher design with implementation considerations in mind. Sections 3 and 4 respectively describe the impact that the number of cipher variables and state size have on the cipher's speed. Section 5 shows how to choose cipher operations for maximum speed. Section 6 describes how special features of the Intel Pentium 4 may be useful in implementing ciphers. Section 7 contains a discussion and summary of results.

2 Grounding Cipher Design in Reality

Many cipher designers hold the misconception that they can derive an accurate estimate of their cipher's efficiency by counting the number of its operations. This belief is reinforced by the prediction of Schneier and Whiting [15] in 1997 that all personal computing processors were converging rapidly to a RISC (Reduced Instruction Set Computing) architecture. History has shown this trend to be mythical. The Pentium 4 architecture is very complex, and this way of obtaining a benchmark on the cipher is inaccurate.

Consider the MAG cipher specification [16], which computes the relative efficiency of that cipher to RC4 by counting the number of operations in their update functions. It is concluded there, without the support of empirical evidence, that 32-bit MAG is four times faster than 8-bit RC4, due to the number and nature of their respective operations. The results given in Section 7 show that a basic implementation of MAG is more than four times slower than a basic implementation of RC4. Similar counting strategies are employed for HERMES [12], the estimate of which is at least qualified by specifying an 8-bit RISC architecture, and for MIR-1 [14], which also mistakenly assumes a lower bound of one operation per clock. Counting operations is not an effective method for calculating cipher efficiency.

Guideline 1 **(High).** Run practical tests during the design process to gauge the efficiency of the cipher.

A corollary to this is "don't make up concrete timing information based upon your paper design". The reality of a cipher's efficiency is connected not just with the contained operations, but also with how well its design is matched to the target architecture. This means that the performance of a cipher will vary even between variants of a processor (for example, the Willamette, Northwood and Prescott variants of the Intel Pentium 4).

For example, MUGI [17] looks simple on paper: it uses an Linear Feedback Shift Register (LFSR), s-boxes, byte swapping, and simple arithmetic operations. However, it runs slowly on the Pentium 4, at around 25 cycles/byte [8], being a 64-bit cipher implemented on a 32-bit platform. The simple operation counting outlined above does not determine the impact of this design-architecture mismatch. The story is similar for 64-bit MIR-1 [14]. The second guideline not only leads to a better approximation of the performance of the cipher, but also to a more efficient cipher.

Guideline 2 **(High).** Understand (in general terms) the architectures on which the cipher is likely to be implemented.

By paying attention to the Intel register set, how data is transferred between the registers and memory, and how it is processed by the CPU, the cipher designer can learn how to make a more efficient cipher. The designer can also manipulate architecture-specific features to this end.

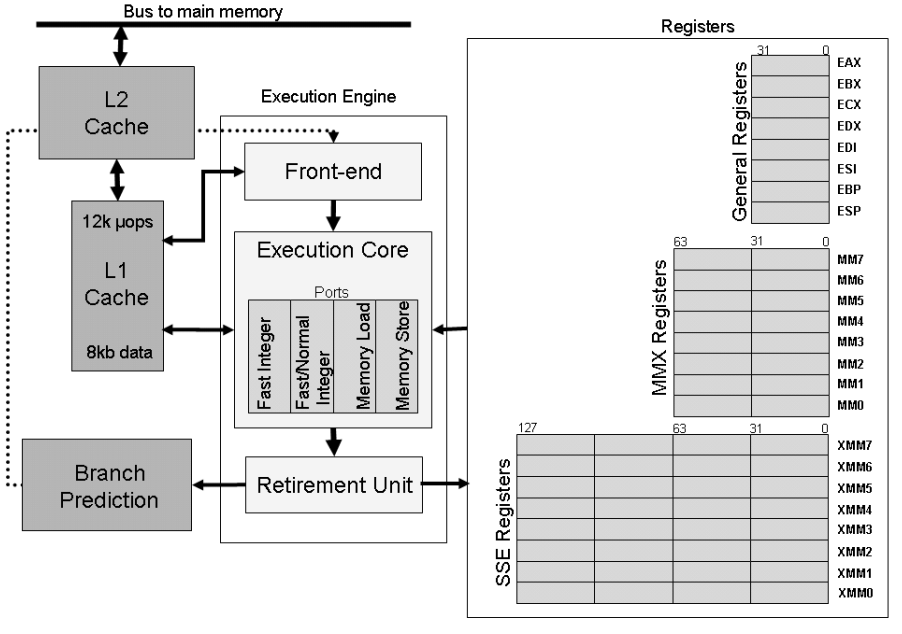


Fig. 1. Intel Pentium 4 Architecture

3 The Pentium 4's Register Set

Registers are very fast memory locations on the CPU die that operate at clock speed, rather than the slower speeds of general-purpose memory. Ideally, each variable in the cipher algorithm can be mapped directly to a register, rather than to general-purpose memory. The Intel Pentium 4 is register-poor in that the programmer has direct access to only eight 32-bit general purpose registers, shown at the right of Figure 1.

Not all registers are treated equally: for example, the multiplication instruction MUL works exclusively upon the EAX and EDX registers; common string operations operate on ESI and EDI; and stack-based operations relating to procedure calls and local variables use EBP and ESP. This reduces the availability of these registers without precluding their use. The instructions that operate on these eight registers, and the rules for dealing with them are found in [9].

Most members of the the Pentium family sport additional registers that can also be used by the programmer. The Multimedia Extensions (MMX) duplex a series of eight 64-bit registers on the back of the Floating Processor Unit (FPU). Members of the family from Intel Pentium III onwards contain eight independent 128-bit (XMM) registers as part of the Streaming Extensions (SSE). The MMX and XMM registers are controlled by their own sets of specialized (and in most regards, limited) instructions. So there are three sets of registers, and three sets of mostly incompatible instructions.

Register pressure occurs when, at a given time, there are more cipher variables to deal with, than there are registers in which to store them. In this case, some of the registers need to be stored in slower memory. This is likely to have a negative impact on the performance of the cipher.

Guideline 3 **(Medium).** For optimal efficiency, the number of variables in any locality of the cipher algorithm should not exceed the number of usable registers.

A good example of adherence to this guideline can be seen in the stream cipher Phelix [18]. Its designers chose to use a 160-bit internal state in their cipher, because this directly maps to five 32-bit registers, meaning that the state does not need to be stored in slower memory. In Mir-1, register pressure is generated through the use of the four sixty-four bit variables in its *loop_update_state* function. These variables translate into eight thirty-two bit variables x_i^t, $0 \leq i < 8$. Because each updated variable x_i^{t+1} depends upon all x_i^t, eight additional registers are required. Ignoring the intermediates, this results in a demand for ten registers at any time. This problem is insoluble using just the general purpose registers.

4 Transferring Data

The Pentium 4's registers are accessible at clock speed (between 1.3 GHz and 3.8 GHz). The memory from which the registers acquire their data is up to three

times slower, and there is a latency penalty of as much 75 cycles, depending upon the chip variant, the front side bus, and how the data transfer is managed. The Pentium 4 contains mechanisms to reduce memory bottlenecks, one of which is a series of high-speed *caches* positioned between the CPU and the main memory. Data is summoned, on demand, to the caches from the main memory. When the data is required by the CPU, but not available within the registers or the caches, then a efficiency penalty (hidden to the cipher designer) is incurred.

The Pentium 4 generally has two caches: the first, the L1 cache, operates at clock speed. In the Willamette and Northwood, the L1 cache has a data capacity of eight kilobytes. In the Prescott, this capacity is increased to sixteen kilobytes. The L2 cache has a shared code and data capacity of between 256 and 2,048 kilobytes. There is L1 cache access latency of between two and four cycles, which increases to as much as sixteen cycles for the Willamette/Northwood's L2 cache (twenty-three cycles for the Prescott). These latencies are incurred if the data is not present in the registers, and has to be summoned from the cache.

The L1 cache stores up to 12,000 micro-operations (μops), which are decomposed assembly code instructions. Most instructions are converted into between one and four μops. Programs which contain more than 12,000 μops are partially stored in the L2 cache or memory. While manipulating the caches is the task of the implementer rather than that of the cipher designer, there is a simple guideline that the designer can use to benefit the implementer.

Guideline 4 **(Medium).** Design the cipher so that both its code and the state fit within the L1 cache.

This guideline is ranked as Medium, because a cipher that does not fit within the L1 cache is still shielded from the effects of slow memory by the much larger L2 cache. Phelix [18] adheres to this guideline, with its 160-bit state and small code size easily fitting into the Northwood's L1 cache. The block cipher LOKI97 [2] uses two s-boxes, one 13×8 and the other 11×8. Together this amounts to ten kilobytes of memory, and the s-boxes will not fit into the Willamette/Northwood's L1 cache. The tabular form of the AES also causes pressure in the L1 cache, as noted by its degraded performance [20].

One of the more interesting examples in this regard is Py [1]. It is one of the fastest ECRYPT ciphers on the Pentium III, with a throughput of one byte for each 2.8 cycles. Py uses two large tables. Elements in each table are modified using elements from both tables. The "novelty" of Py is the *rolling array*, which uses an old optimization trick commonly applied to LFSRs.

The traditional way to represent a LFSR with l stages is to use a circular buffer B of size $m = l$. A pointer $p, B \leq p < B + m$ indicates the location of stage W_0. The element W_1 is located at $p+1$ if $p+1 < l$ otherwise at location B. When the LFSR clocks, the feedback is written into the LFSR (usually at W_0), and the pointer increments by one. No elements are moved, or changed, except for where the feedback is placed. At each step, the pointer has to be checked to see if p equals $B + m$, in which case it is set to B. Any access into the LFSR also need to be bound checked.

Py's designers reason that the check on p is an expensive overhead. Py mitigates this expense by trading off memory. The trick it uses to allocate a buffer that is much larger than the LFSR, that is, $m \gg l$. As time passes, the pointer p moves along this buffer, and the feedback element normally written at W_0 is instead written at W_l. When the pointer reaches the end of the buffer, all of the elements representing the LFSR, from $B+m-(l+1)$ to $B+m-1$ are physically copied back to B through to $B+l$. The bound check on p needs to be made once in $(m/|W|)$ words of outputs, where $|W|$ is the size of the output word. This strategy can be adapted to any cipher in which the feedback is written only to the contiguous positions at the start of the LFSR. The copy operation employed by Py is expensive, but the cost is amortized over all of the intervening operations. Thus Py is at its most effective when the size of the buffer approaches the size of the L1 cache.

The default size for the buffers in the Py code submitted to ECRYPT is $b = 4000$ stages, where each stage is 32 bits. There are two buffers, so the state size is 32 kilobytes, which is commensurate with the size of the Pentium III's L1 cache. Clearly the cipher is less efficient on the Pentium 4, since the copy operation needs to be called four times more frequently to accommodate the much smaller size of that machine's L1 data cache. Benchmarks for Py with varying buffer sizes are given in Table 1 in Section 7.

5 Processing Data

From the cache, the data moves to the CPU. The cipher designer needs to understand how the data is processed, and from there choose instructions that optimize that processing.

5.1 The Pentium 4's Execution Engine

The CPU operates on data summoned to the registers, using its three-phase execution engine. The first phase, the Instruction Front-End converts assembly instructions into μops and supplies them in the order provided through the instruction portion of the L1 cache to the Instruction Execution core.

In theory, the *super-scalar* Instruction Execution core is able to execute six μops (for example, six exclusive-ors) within a single clock-cycle. It re-orders the instructions provided by the Instruction Front-End to provide the fastest possible execution, withstanding dependencies between the operands. The core has four ports that provide access to its execution units, including

1. port #0 has a fast integer unit and a MMX/SSE move unit[2]. The port can dispatch two fast integer operations per cycle; or one fast integer operation and one MMX/SSE move or store operation. Fast integer operations include logical operations, addition, subtraction and exclusive-or.

[2] Floating point operations are ignored in this discussion, since floating point arithmetic rarely features in symmetric cipher design.

2. port #1 has a fast integer unit, and a normal integer unit. The port can dispatch two fast integer operations per cycle. An integer multiply, shift or rotate, or MMX/SSE operation can be substituted for the second fast integer operation.
3. port #2 deals with fetching memory for the general purpose registers. It can deal with one operation per cycle.
4. port #3 deals with storing data to memory. It can deal with one operation per cycle.

When the execution units have processed the μops, the third phase of the execution engine, the Instruction Retirement Unit, takes them and retires them in the correct order to maintain program correctness. Although the instruction core can execute six μops in parallel, the instruction retirement unit retires three μops in one clock cycle. This is the upper bound on the CPU throughput.

Guideline 5 **(Medium).** To take advantage of the Pentium 4's super-scalar ability, ensure that all instruction ports are occupied at all times for optimal execution.

There are some mechanisms within the Pentium 4 architecture, such as register renaming, which work to maximize the efficiency of the core (which is why this guideline has a medium priority). But the cipher designer needs to consider the types and order of operations used within a cipher. For example, if the cipher design requires two parallel executions of a multiplication, they will be executed serially as only port #1 handles general-purpose multiplication. Alternatively, if the cipher design requires parallel execution of a multiplication and an addition, the operations can be shunted to ports #1 and #0 respectively for simultaneous execution.

5.2 Throughtputs and Latencies

Each instruction is associated with a *latency* and a *throughput*. The latency of the instruction is the delay incurred before the CPU can operate on the next dependent instruction. For example, the latency affects the timing of the instruction sequence $a = a \oplus b; a = a \ll c$. The throughput is the delay before an independent operation can be scheduled on the same execution unit. For example, throughput affects the timing of $a = a \oplus b; c = c \oplus d$.

On the Intel Pentium 4, fast integer operations such as 32-bit addition, subtraction, exclusive-or and logical operations such as AND and OR all have throughputs and latencies of $\frac{1}{2}$ cycle (except on the Prescott, where these operations have latencies of one cycle). Two of these $\frac{1}{2}$ cycle operations can be scheduled and completed on a fast integer port within each clock cycle (assuming that their operands are dependency-free and located within the registers). As there are two fast integer ports, this means that up to four fast operations can be processed within the execution unit per cycle! This is a peak rate, and is subject to the limiting factor of three operations per cycle on the Instruction Retirement unit.

Shifts and rotates are executed on the normal integer unit, and on the Willamette/Northwood, have a worst-case latency of four cycles. On the same platforms, multiplication has a worst case throughput on five cycles, and latency of up to eighteen cycles. Timings for all operations can be found in [10].

Many 32-bit compilers offer decent support for 64-bit operands, and translate simple 64-bit arithmetic and logical operations into the underlying 32-bit instruction codes quite efficiently. A sixty-four bit exclusive-or can be simulated using two thirty-bit exclusive-ors. A sixty-four bit addition can be simulated using two thirty-bit additions, a comparison, and possibly an additional carry. Sixty-four bit rotation can be simulated using four thirty-two bit shifts and two exclusive-ors. On a per-bit basis, none of these operations is much worse than on 32-bit operands.

Mir-1's *loop_state_update* sub-function updates four sixty-four bit variables using the simple operations of addition, shifting, AND, OR, and multiplication. A sixty-four bit multiplication can be simulated on thirty-two registers using four multiplications, and eight additions. Considering the dependencies between operations, the four multiplications in *loop_state_update*, responsible in the production of one eight byte keystream word, expend as much as one hundred and twelve clock cycles!

***Guideline 6* (High).** Choose operations with low latency and low throughput. Avoid expensive operations such as multiplication or division, unless necessary.

The fastest ECRYPT ciphers on this platform (HC-256 [19], Phelix [18], and Py [1]) adhere to this guideline and also do not make extensive use of s-boxes or table-lookups. Unlike other cipher primitives, s-boxes do not take single or fractional clock cycles. For example, a single 8×8 s-box lookup $y = S(x)$ on a 32-bit machine may take up to three operations, as shown in Figure 2.

```
movzx eax, DWORD PTR _x$[ebx]     ; copy source to index register
mov   ecx, DWORD PTR _sbox[eax*4] ; copy address of s-box
mov   DWORD PTR _y$[ebx], ecx     ; perform s-box lookup
```

Fig. 2. Assembly Code for 8×8 S-box Lookup

Because of the dependencies between them, these operations all occur in serial. Register pressure prevents complete parallelization of more than two neighbouring s-box lookups. Additionally, the cache issues demonstrated in Section 4 come into play.

***Guideline 7* (Medium).** S-boxes are large and slow. Use with caution.

This guideline has a Medium priority because the s-box provides a potentially rich source of non-linearity. Many ciphers, including HERMES [12], use 8×8 s-boxes, whereas Dragon [3] uses 32×32 s-boxes. The use of larger s-boxes improves their non-linearity, but also poses a practical problem: whereas 8×8 s-boxes require 256 bytes of storage, 32×32 s-boxes require sixteen gigabytes.

This is clearly not practical, so in Dragon a virtual s-box is constructed as $y = S_0(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_3(x_3)$ where $x = x_0 \| x_1 \| x_2 \| x_3$. The memory requirement of the virtual s-box is further reduced to two kilobytes by reusing one of the 8×32 s-boxes, avoiding a likely L1 cache overflow. Performing a lookup on Dragon's virtual 32×32 s-box consists of four 8×32 s-box lookups, isolating individual bytes in the source word, using three ANDs and three shifts, and combining the results using three exclusive-ors. Four index registers are required for a single 8×32 s-box. As there are six s-box implementations, and seven registers, register pressure means that the s-box invocations are compelled to be serialized. This was considered by the designers of Dragon as a necessary security efficiency trade-off.

Guideline 8 **(High).** Maximize productivity for processor work.

Having chosen a series of efficient operations that provide the cipher with necessary security, make sure that the work performed by processor to execute those operations is not wasted. The most obvious way to ensure this is to output keystream in blocks that are multiples of the word size used by the cipher's internal state. HC-256, Py, RC4, and MIR-1 all output keystream blocks that are same size as an internal state word. Dragon has a 32-bit internal state word, and outputs 64-bit words. These all represent good usage of the work performed by the processor. Conversely, MAG outputs a block one-quarter the size of the internal state word, which wastes much of the work performed by the processor.

5.3 Branch Prediction

Some ciphers use branching as an intrinsic part of the algorithm. For example, MAG's concise update function contains just four assignments, including one based upon the result of an inequality comparison (that is, a branch).

The Intel Pentium 4 has a branch prediction unit that analyzes the anticipated path of the branch, executes it, and prepares the result by the time the branch expression is evaluated. When the branch is mis-predicted, the results are discarded and many cycles of CPU time are wasted, depending upon the size of the execution pipeline. The Pentium 4 has an exceptionally large pipeline, with as many as 31 stages, compared to the ten stages of the Pentium III. Thus a branch mis-prediction will have much worse consequences on the former. Unfortunately, the branch prediction algorithms of the Pentium 4 perform badly with random data, causing a delay of up between 24 and one hundred cycles with each mis-prediction. This situation is worse than if the Pentium 4 contained no branch prediction at all [7].

Guideline 9 **(Medium).** Avoid branching within the cipher algorithm.

Branch mis-prediction plays a major role in the poor performance of MAG. Yet RC4, to which MAG is most closely compared by the latter's designer, contains no branching of this type. The small size of MAG's update function means that the CPU has no chance to recover from the mis-prediction penalties. This is one

reason as to why, despite the similar number of operations, MAG is four times slower than RC4. If branches can not be omitted, in some cases the pipeline penalties can be avoided by careful coding (for example, using the CMOV or SETcc instructions).

6 Instruction Extensions

The MMX and SSE extensions have potential as a valuable avenue for cipher implementors. The MMX extensions offer 47 new instructions on eight 64-bit registers. The SSE2/SSE3 instructions bring a wide range of integer and floating point operations to a set of eight 128-bit XMM registers. Many of the instructions are variations based around the size of the operands, which can have widths of 8, 16, 32, 64, or (for SSE) 128 bits. The primary benefit of MMX/SSE, aside from the additional registers, is that one instruction processes multiple operands in parallel.

The new instruction sets are somewhat deficient, and not particularly efficient. The SSE instructions have typical throughputs and latencies of two cycles. This puts them on an even footing with the fast integer operations on general purpose registers, which operate on 32-bits and execute in half a cycle.

Most valuable to cryptographers are the instructions for parallel execution of simple arithmetic and logical operations, and for a faster multiplication. The MMX and SSE instructions cannot address memory, which poses problems with s-boxes. S-boxes can be implemented on the general purpose registers, and indexed using operands computed using MMX/SSE. There is a strong penalty of six cycles for transferring data between the XMM registers and general purpose memory, making it problematic to implement any cipher containing s-boxes using MMX/SSE. Also the MMX and SSE instruction sets lack branching instructions, which can be simulated at a price on the general purpose registers.

Guideline 10 **(Medium).** Do not assume that there is a significant advantage in using MMX or SSE instruction extensions.

In particular, stream ciphers based on small functions that incorporate s-boxes, or other indirect addressing mechanisms, are probably not amenable for implementation using MMX or SSE. Some of the ECRYPT ciphers that are not suitable for efficient implementation using these extensions include Dragon, Py and HC-256.

7 Discussion and Conclusion

In symmetric cipher design, security is paramount and efficiency is a secondary consideration. However, it is an area in which there are now many successful candidates, fulfilling both security and efficiency requirements. The ciphers are judged now, on the platforms that we have available. A cipher designer who does not understand at a high-level, the architecture of the target machine, should expect sub-optimal performance from the cipher. This results in a non-competitive cipher.

Table 1. Performance of Modern Stream Ciphers on the Intel Pentium 4 (Northwood)

Cipher	Word Size (bits)	Internal State (bits)	Features	Payload (bytes)			
				100	1,000	10,000	10^9
				Throughput (mbits/s)			
Dragon	32	1,088	large s-boxes	407	708	1,028	1,380
HC-256	32	65,536	very large tables	7	66	414	1,372
MAG	32	4,096	frequent branching	< 1	7	43	163
MIR-1	64	768	64-bit multiplication	198	362	419	464
Py-1	32	1,560	no rolling array	59	258	491	602
Py-4	32	6,240	large state	68	270	443	797
Py-16	32	24,960	very large state	65	256	446	801
RC4	8	2,048	small and fast	244	488	622	729

The performance of some sub-optimized stream ciphers is shown in Table 1, for varying payload lengths. The metrics are gathered from an Intel Pentium 4 (Northwood) processor, running at 2.6 GHz, with an 8 kilobyte L1 data cache. For each test, one gigabyte of data is encrypted. When the specified payload has been encrypted, IV rekeying occurs. This overhead is incorporated into the metric, which is expressed as megabits per second. Py-x represents Py with a rolling array $x\times$ the state size of Py implemented using traditional techniques.

Those ciphers that most closely adhere to the guidelines issued in this paper — Py, Dragon, and HC256 – are the fastest on the Intel Pentium 4. The metrics of all the ciphers shown degrade with frequent rekeying, but those worst affected are those that rekey by updating each element in a large state (such as HC-256), or by excessively updating each element (for example, MAG, which during rekeying discards 2^{14} bytes of keystream, for each 100-byte payload). These metrics imply that designers need to pay attention to the key agility of their ciphers. They also suggest that the rolling arrays of Py lose their advantage on architectures with small cache sizes, such as the Intel Pentium 4.

This paper does not suggest that ciphers should be designed exclusively for the Intel Pentium 4. Most of the guidelines should be considered irrespective of the targeted architecture. All that is required to implement the guidelines is some knowledge about computer architectures, including the register set; the memory layout including the size and speed of the caches; the types, throughputs and latencies of available instructions; and special features of the architecture, such as SSE2.

References

1. E Biham and J Seberry. Py (Roo): A Fast and Secure Stream Cipher Using Rolling Arrays, 2005. Available at http://www.ecrypt.eu.org/stream/py.html.
2. L Brown and J Pieprzyk. Introducing the new LOKI97 block cipher. In *First Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology (NIST)*, August 1998. Available at http://csrc.nist.gov/encryption/aes/.

3. K Chen, M Henricksen, L Simpson, W Millian, and E Dawson. Dragon: A fast word based cipher. In *ICISC '04*, volume 3506 of *LNCS*, pages 33–50, 2004.
4. J Daemen and V Rijmen. Rijndael. In *First Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology (NIST)*, August 1998. Available at http://csrc.nist.gov/encryption/aes/.
5. ECRYPT. Call for stream cipher primitives, April 2005. Available at http://www.ecrypt.eu.org/stream/call/.
6. ECRYPT. ESTREAM End of Phase 1, March 2006. Available at http://www.ecrypt.eu.org/stream/endofphase1.html.
7. A Fog. How to optmize for the Pentium family of microprocessors, 2004. Available at http://www.agner.org/assem/pentopt.pdf.
8. M Henricksen and E Dawson. Rekeying issues in the MUGI stream cipher. In *SAC'2005*, volume 3897 of *LNCS*. Springer-Verlag, 2006.
9. Intel Corporation. *IA-32 Intel Architecture Software Developers Manual Volume 2: Instruction Set Reference*. Intel Press, 2001.
10. Intel Corporation. *Intel Pentium 4 and Intel Xeon Processor Optimization Reference Manual*. Intel Press, 2001.
11. C Jansen, T Helleseth, and A Kholosha. Cascade Jump Controlled Sequence Generator (CJCSG), 2005. Available at `http://www.ecrypt.eu.org/stream/ciphers/pomaranch/pomaranch.pdf`.
12. U Kaiser. Hermes Stream Cipher, 2005. Available at `http://www.ecrypt.eu.org/stream/hermes.html`.
13. M Matsui and S Fukuda. How to maximize software performance of symmetric primitives on Pentium III and 4 Processors. In *FSE 2005*, LNCS, pages 398–412. Springer-Verlag, 2005. To appear.
14. A Maximov. A new stream cipher "Mir-1", 2005. Available at `http://www.ecrypt.eu.org/stream/mir1.html`.
15. B Schneier and D Whiting. Fast software encryption: Designing encryption algorithms for optimal software speed on the Intel Pentium Processor. In *FSE '97*, volume 1267 of *LNCS*. Springer-Verlag, 1997.
16. R Vuckovac. MAG My Array Generator (a new strategy for random number generation), 2005. Available at http://www.ecrypt.eu.org/stream/mag.html.
17. D Watanabe, S Furuya, H Yoshida, and K Takaragi. A new keystream generator MUGI. In *FSE'03*, volume 2365 of *LNCS*, pages 179–194. Springer-Verlag, 2003.
18. D Whiting, B Schneier, S Lucks, and F Muller. Phelix: Fast Encryption and Authentication in a Single Cryptographic Primitive, 2005. Available at http://www.ecrypt.eu.org/stream/phelix.html.
19. H Wu. A New Stream Cipher HC-256, 2004. Available at `http://eprint.iacr.org/2004/092.pdf`.
20. E Young. Re: More LTC timings... Posting to sci.crypt usenet group on 18 June, 2003.

Improved Cryptanalysis of MAG*

Leonie Simpson[1] and Matt Henricksen[2]

[1] School of Software Engineering and Data Communications,
Queensland University of Technology
GPO Box 2434, Brisbane Q 4001, Australia
lr.simpson@qut.edu.au
[2] Information Security Institute, Queensland University of Technology
GPO Box 2434, Brisbane Q 4001, Australia
m.henricksen@isrc.qut.edu.au

Abstract. MAG is a synchronous stream cipher submitted to the E-CRYPT eSTREAM project. The design criterion for the cipher is cellular automata, although it can be modelled as a word-based shift-register with a single word of memory. Cryptanalysis of MAG reveals serious structural weaknesses within the cipher. This paper presents simple distinguishing attacks against MAG with an 80-bit or 128-bit key that can, under certain circumstances, be carried out by hand. The approach is extended to a partial-key recovery attack. For the 80-bit and 128-bit keys, we recover 40 key bits and 32 key bits respectively from about 32 bytes keystream. A proposed modification to MAG, intended to prevent an earlier distinguishing attack, has no effect upon our distinguisher but instead allows a full key recovery attack for both 80-bit and 128-bit keys using around thirty-two bytes of keystream and a practical precomputation. Therefore the modification actually weakens an already insecure cipher.

Keywords: Cryptography, Stream Ciphers, Key Recovery, Distinguisher, MAG Keystream Generator.

1 Introduction

MAG (My Array Generator) [3] is a synchronous stream cipher submitted to the eSTREAM stream cipher project at the ECRYPT network of excellence. MAG is submitted under both Profile 1 (software-suitable cipher using a 128-bit key) and Profile 2 (hardware-suitable cipher using an 80-bit key) categories.

MAG's design is influenced by cellular automata, in which the state of each cell in the automata changes over time, based upon the state values of its neighbors. MAG is one of a long line of stream ciphers based upon cellular automata, and is strongly influenced by Wolfram's Rule 30 stream cipher [6] from 1985. MAG uses conditional branching rather than algebraic functions to provide the cipher with complexity. MAG's designer Vuckovac says in its specification:

* This research was supported by Australian Research Council Discovery Project Grant DP0450920.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 64–75, 2006.
© Springer-Verlag Berlin Heidelberg 2006

> The advantage of [this] approach is: a set of criteria such as linear complexity, nonlinearity, statistics, confusion and diffusion does not have to be addressed directly as is the case with more complicated system-theoretic approaches. The whole security issue is shifted to the computational irreducibility principle alone. [3, page 15].

In this paper, we show that MAG can be modelled as a cipher consisting of a word-based shift register combined with a single word of memory. This model permits analysis with respect to the set of criteria Vuckovac has not directly addressed. From this perspective, cryptanalysis of MAG reveals structural weaknesses that can be used in a distinguishing attack. When combined with the rekeying scheme, this allows recovery of a substantial portion of the key in a simple known plaintext attack.

This result improves upon the previous cryptanalysis of Künzli and Meier [1]. Their distinguishing attack on MAG requires up to 129 consecutive words of keystream, with negligible operational and memory requirements. Künzli and Meier's attack makes no use of MAG's weak rekeying strategy, and does not recover any key material, however it demonstrates that the cryptanalyst can distinguish between the keystream produced by the MAG cipher and a random bitstream. Some cryptographers regard distinguishing attacks as a far less significant security threat than key recovery attacks [2], so in this paper we also demonstrate simple key recovery attacks on MAG.

Vuckovac proposed modifications to the output function of MAG in response to Künzli and Meier's distinguishing attack. Although the modifications do defeat that particular attack, they do not prevent other distinguishers, such as the distinguisher described in Section 4 of this paper. Additionally, the modifications enable the partial key recovery attack described in this paper to be extended to a complete key recovery attack.

In Section 2, we describe in detail our model of the MAG stream cipher. In Section 3, we make some observations about the stream cipher, including structural weaknesses that can be exploited by cryptographic attacks. In Section 4, we introduce a new distinguishing attack on MAG and detail a partial key-recovery attack. In Section 5, we address a modification made by Vuckovac in [4]. We show that although this modification may provide greater variability in the keystream, increasing the complexity of the Künzli and Meier distinguishing attack [1], it has negligible impact upon our new distinguishing attack, and improves the performance of our key recovery attack upon MAG so that complete key recovery is possible. Section 6 presents a discussion and summary.

2 Description of the MAG Stream Cipher

MAG is based upon a cellular automaton. The MAG submission to ECRYPT [3] contains a vague textual specification, and a reference implementation of the cipher. There are inconsistencies between the two. The description in this section is gleaned from the implementation, which spells out the structure of the cipher.

The internal state of MAG contains a large shift register, denoted R, which has 127 stages each consisting of a 32-bit word. An additional 32-bit word, described as a "carry", acts as the cipher's memory. Thus the total internal state consists of 128 32-bit words, or 4096 bits. Let R_i^t denote the contents of the i^{th} stage of register R at time t, where $0 \leq i < 127$. Let C^t denote the contents of the memory word at time t.

We use the following notations to represent the algebraic operations used by the cipher: addition modulo 2^{32} ($+$), binary addition ($\oplus$), concatenation ($||$), truncation ($trunc_i$), where only the i least-significant bits of the argument are retained, bitwise complementation ($\overline{R}$), and word-based comparison ($<$).

As is standard for modern stream ciphers, MAG's state is populated using a dedicated key initialization algorithm. The state is subsequently modified using an update function, which is used in conjunction with the cipher's output function to produce the keystream. The original MAG specification [3] does not describe any key initialization algorithm. The reference implementation has been used to derive the algorithm described below. This algorithm does not make provisions for initialization vectors (IVs). This causes practical key management problems, particularly with modern cryptographic protocols that use IVs to avoid related-key attacks. Although the eSTREAM project profiles request stream ciphers that use IVs, in this paper we analyse MAG as it has been specified within the code. Note that earlier this year, Vuckovac released a note on the cipher design [5] which addresses the use of IVs by treating them as a supplement to the key. That is, the key and the IV are concatenated, and the resultant is used in the manner described for keys in the algorithm below. For example, for an 80-bit key and a 64 bit IV, this would be concatenated and treated as a 144-bit key (although 64 bits are known). Thus the approach described for keys without IVs in this paper can be applied directly when IVs are used, and may be more successful.

2.1 MAG's Update Function

The update function of MAG is used by both the output function and the key initialization algorithm. The update function takes five 32-bit words as input: the carry element and four consecutive stages from start of the shift register R. The update function is performed in two phases, as follows.

Firstly, the carry element C is modified, using the result of a comparison between two stages in the register.

$$C^{t+1} = ((C^t \oplus D^t) + E) \bmod 2^{32} \tag{1}$$

$$\text{where } D^t = \begin{cases} R_1^t \text{ if } R_2^t > R_3^t \\ \\ \overline{R_1^t} \text{ if } R_2^t \leq R_3^t \end{cases} \tag{2}$$

and E is an arbitrary 32-bit constant when used in the key initialization, or zero when used in keystream generation. D represents the branching component of the cipher.

Secondly, register R is updated as follows:

$$R_i^{t+1} = \begin{cases} R_i^t \oplus C^{t+1} \text{ for } i = 126 \\ \\ R_{i+1}^t \qquad\quad \text{for } 0 \leq i < 126 \end{cases} \tag{3}$$

The update function is depicted in Figure 1.

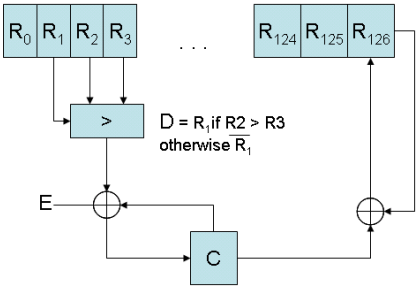


Fig. 1. MAG update function

2.2 MAG's Key Initialization

MAG's key initialization function is two-phased. In the first phase, the initial state is populated using a key K, and in the second phase, the state is modified using the MAG update function.

The first phase of the initialization algorithm uses a key K of length l bits, where $0 \leq l \leq 4,096$ and generates an extended key EK of length 4,096 bits by concatenating copies of the key K. Let k_i represent the i^{th} of K, where $0 \leq i < l$. Then, K is represented by $K = (k_0 k_1 \ \ldots \ k_{l-1})$, and

$$EK = (k_0 k_1 \ \ldots \ k_{l-1} \parallel k_0 k_1 \ \ldots \parallel k_{4,096 \bmod l})$$

This extended key EK forms the initial state of the cipher. That is,$EK = R_0^0 \| R_1^0 \| \ldots \| R_{126}^0 \| C^0$ at the start of the second phase of initialization.

As MAG has been submitted to the eSTREAM project under profiles 1 and two, it is expected to support both 80-bit and 128-bit keys. In the remaining sections of the paper, we discuss attacks for these specific key sizes, without IVs, so here we define notation relating the 32-bit words of the initial states and specific key bits for these two key sizes.

A 128-bit key can be represented as a sequence of four 32-bit subkeys $S_i, 0 \leq i \leq 3$. Using this notation, the extended key can be represented by the concatenation of these four subkeys, with each subkey appearing thirty-two times, as follows.

$$EK = (S_0 \parallel S_1 \parallel S_2 \parallel S_3 \parallel ... \parallel S_0 \parallel S_1 \parallel ... \parallel S_3 \ ) \text{ where}$$
$$S_0 = (k_0, k_1, ..., k_{31}), \ S_1 = (k_{32}, k_{33}, ..., k_{63})$$
$$S_2 = (k_{64}, ..., k_{95}), \ S_3 = (k_{96}, k_{97}, ..., k_{127})$$

An 80-bit key is not a multiple of the 32-bit word size. However, the extended key formed from concatenation of an 80-bit key can be represented as the concatenation of a sequence of five 32-bit subkeys $S_i, 0 \leq i \leq 4$. These are simply the five 32-bit words which are formed when two copies of the 80-bit key are concatenated. Note that particular key bits appear in different positions in the subkeys.

$$\begin{aligned} EK &= (S_0 \parallel S_1 \parallel ... \parallel S_4 \parallel S_0 \parallel S_1 \parallel ... \parallel S_4) \text{ where} \\ S_0 &= (k_0, k_1, ..., k_{31}),\ S_1 = (k_{32}, k_{33}, ..., k_{63}) \\ S_2 &= (k_{64}, ..., k_{79}, k_0, ..., k_{15}), S_3 = (k_{16}, k_{17}, ..., k_{47}) \\ S_4 &= (k_{48}, ..., k_{79}) \end{aligned}$$

In the second phase of key initialization, the state is modified by iteratively calling the update function F times, using an arbitrary 32-bit constant E. After this has occurred, the cipher is ready to produce keystream.

In the reference code, the constant E is assigned the hexadecimal value 0x11111111. A comment within the code indicates that this value is chosen arbitrarily. In the attacks described in this paper, we assign the value of zero to the constant, the same value that is used during keystream generation. As this is an arbitrarily chosen but known constant, applied in every iteration of the initialisation phase, it is our belief that the chosen value should not be critical to the security of the cipher.

The value assigned to F by the reference code is 2^{14}; that is, the key initialization algorithm iterates the update function 2^{14} times. From an efficiency perspective, this amount seems excessive. For applications which specify frequent rekeying (for example, at the end of each frame), the key initialization in MAG may be as much as an order of magnitude slower than the bulk encryption of the frame. From the perspective of cryptanalysis, when the value of constant E is chosen to be zero, then the value of F has no effect on the requirements of our attacks.

2.3 MAG's Output Function

Following initialization of the cipher, the output function generates one byte of keystream after each clocking of the register. The keystream byte z^t produced at time $t, t \geq 1$ is generated by truncating the contents of the last stage of the register R. That is,

$$z^t = trunc_8(R_{126}^{t+1})$$

3 Observations on Weaknesses in MAG

In this section, we make a series of observations on MAG about structural weaknesses that can be exploited in both distinguishing and key recovery attacks.

1. MAG has a large internal state, with a 4,064 bit register R, consisting of 127 $\times$ 32-bit words. The update function clocks the register once, altering a single word. No other diffusion mechanisms are present. Consequently diffusion

between words within the register is particularly slow. A single change made to one of the stages will be reflected in all other stages in the register only after 127 calls to the update function have been made. The cryptanalyst has more opportunity to amass information about the internal state when it is mixed slowly.

2. Of the 127 stages in the register R, only the first four are considered in the generation of any word of keystream.
3. For an instance of the cipher initialized with a key longer than 128 bits, a guess-and-determine attack is possible by guessing the contents of R_0, R_1, C, and one bit relating to whether $R_2 > R_3$. For keys much larger than 128 bits, such an attack will be particularly effective.
4. During keystream generation, the only mixing operation used is exclusive-or. This bitwise operation provides no diffusion within 32-bit words. During keystream initialization, modular addition of a constant of E is also used, but this provides very limited diffusion.
5. The output function takes the least significant byte of one register stage and, without any filtering, emits it as keystream. Thus the keystream provides a window into the internal state. Note that after the register has been clocked 127 times, one quarter of the internal state has been revealed. Given the weakness of the update function, this should reveal a large amount of key material.
6. Much is made by Vuckovac of MAG's use of conditional branching (termed in that document as "selection programming structure") rather than boolean or algebraic techniques. He says:

   > MAG evolution uses binary branching through iteration. Therefore [the] complexity of path [through] which one cell evolves increases exponentially. From [the] software testing point of view, [the] MAG algorithm is unmanageable. [3, page 6].

   This claim of unmanageability is demonstrably false, as is the implication that MAG belongs to an unusual paradigm of ciphers, since the update function of MAG can be written in standard algebraic notation. For $E = 0$, as it is for keystream generation, equation 2 can be rewritten as:

$$C^{t+1} = C^t \oplus \sum_{i=0}^{t} R_{126}^{t+1} \oplus G(N^t \bmod 2) \tag{4}$$

   where $G = (-1) \bmod 2^{32}$ and N^t is equal to the number of times for which $R_2^i > R_3^i$, for $0 \leq i < t$. This simple equation occurs because consecutive complements in equation 2 cancel: for register stages R_a and R_b, $R_a \oplus (-1 \bmod 2^{32}) \oplus R_b \oplus (-1 \bmod 2^{32}) = R_a \oplus R_b$, irrespective of the values of R_a and R_b. Using this representation, it is clear that even if the value of N^t is unknown, that the branching technique used in MAG adds only one bit of uncertainty to each of its output words.
7. During keystream generation, the update function uses only the exclusive-or and complement operations. Therefore, new register words can only be

linear combinations of register words which are present at the end of the initialisation process. If $E = 0$ during initialisation, then new register words can only be linear combinations of the register words formed by loading the key (and the known IVs).

8. The implication of observations 5 and 7 is that irrespective of how many times the shift register R is clocked during key initialization, for some key lengths, there are byte patterns that can never occur in the keystream. Table 1 shows the maximum number of possible byte patterns for keys of lengths between 64 and 128 bits in multiples of eight bits, given $E = 0$. Note that for keys which result in subkeys that are not distinct, the number of distinct output bytes will be fewer than this.

   For a 128 bit key with no IV, there are four 32-bit subkeys. New register words are a linear combination of these subkeys, and the constant G:

$$a_0S_0 + a_1S_1 + ... + a_4G$$

   So the maximum number of distinct output bytes, obtained when the four subkeys are distinct, is 32. For an 80-bit key with no IV, there are five subkeys $S_0...S_4$, and similarly the maximum number of distinct output bytes is 64. For a 128 bit key with a 64-bit IV, this could be considered as a 192 bit key. Thus there are six 32-bit subkeys, although two of these (formed from the IV) are known. New register words are a linear combination of these subkeys, and the constant G:

$$a_0S_0 + a_1S_1 + ... + a_6G$$

   So the maximum number of distinct output bytes, obtained when the four subkeys are distinct, is 128. For an 80 bit key with a 64-bit IV, this could be considered as a 144 bit key. Thus there are nine 32-bit subkeys, although three of these (formed from the IV) are known, and half the bits are known in two other subkeys. New register words are a linear combination of these subkeys, and the constant G:

$$a_0S_0 + a_1S_1 + ... + a_9G$$

   So the maximum number of distinct output bytes, obtained when the four subkeys are distinct, is 256.

   Note that the 80 bit key, when concatenated to form the extended key, produces a larger number of subkeys than the 128-bit key. This results in exposure of a greater number of key bits through the output function. For example, the 128-bit key has four subkeys, from which the output function exposes 32 bits of key material. The smaller 80-bit key has five subkeys, from which the output function exposes 40 bits of key material.

4 Two Attacks on MAG

This section presents a simple distinguishing attack against MAG, considered as in the original submission, without IVs. The approach is then developed into a partial key recovery attack.

Table 1. Maximum number of byte patterns in MAG keystream output

Key size (bits)	max # byte patterns
64	8
72	256
80	64
96	16
128	32
144	256
192	128
256	128
512	256
1024	256
2048	32
4096	32

4.1 A Distinguishing Attack

A successful distinguishing attack against the cipher MAG is able to differentiate between a random sequence and keystream produced by MAG. Künzli and Meier present such an attack in [1] where they observe a relation between keystream bytes $K_i, K_{i+i}, K_{i+2},$ and K_{i+127} for all i. Knowledge of these four bytes allows keystream byte K_{i+128} to be predicted in a MAG keystream as either one of two possible values, with 50% accuracy. The second guess can be made with negligible effort for 100% accuracy. If the actual keystream byte does not match either of these two predicted values, the keystream has not been produced by MAG. Alternatively, a random keystream generator will produce either of the predicted bytes with total probability $\frac{1}{128}$. Two bytes of keystream produced by MAG and the random generator will coincide with probability $\frac{1}{65,536}$. Consequently, Künzli and Meier have presented a simple and effective attack that works with 99.9% accuracy, if only seven bytes of keystream with indices $i, i+1, i+2, i+3, i+127, i+128, i+129$ are known. For 129 consecutive known keystream bytes, their distinguisher works in all cases.

In this paper, we present a distinguishing attack without the requirement that the attacker knows keystream bytes with particular indices. In our attack, knowledge of a handful of keystream bytes with any indices suffices to distinguish the MAG keystream from a keystream produced by a random generator. However, our attack works better for some key lengths than for others, unlike the attack of [1] which works effectively without consideration of MAG's key initialization algorithm. We make the additional assumption that the value of the initialisation constant $E = 0$.

In our attack, we carry out a pre-computation for the given key length to generate a map of outputs in terms of the unknown sub-keys. For example, consider a 128-bit key $K = S_0 \| S_1 \| S_2 \| S_3$. Then we know that following phase 1 of the key initialization algorithm, $R^0 = (S_0 \| S_1 \| S_2 \| S_3 \| S_0 \| S_1 \| S_2)$ and $C^0 = S_3$. Aside from the actual values of the subkeys, the only information that is not

known is the values of $G(N^t \bmod 2)$ (see observation 6). Using equations 3 and 4, we can map out the internal state to the register update in terms of $S_i^t, 0 \leq i < 4$ at any time t, as shown in Table 2. Recall from observation 5 that at time t, the output byte z^t is related to register R_{126}^{t+1}.

Table 2. MAG Keystream phrased in terms of subkeys for $E = F = 0$ and $l = 128$

time (t)	R_1^t	R_2^t	C^{t+1}	R_0^{t+1}
0	S_0	S_1	$S_3 \oplus S_1 \oplus G(N^0 \bmod 2)$	$S_3 \oplus S_1 \oplus S_0 \oplus G(N^0 \bmod 2)$
1	S_1	S_2	$S_3 \oplus S_2 \oplus S_1 \oplus G(N^1 \bmod 2)$	$S_3 \oplus S_2 \oplus G(N^1 \bmod 2)$
2	S_2	S_3	$S_2 \oplus S_1 \oplus G(N^2 \bmod 2)$	$S_1 \oplus G(N^2 \bmod 2)$
3	S_3	S_0	$S_2 \oplus S_1 \oplus S_0 \oplus G(N^3 \bmod 2)$	$S_3 \oplus S_2 \oplus S_1 \oplus S_0 \oplus G(N^3 \bmod 2)$
4	S_0	S_1	$S_2 \oplus S_0 \oplus G(N^4 \bmod 2)$	$S_2 \oplus G(N^4 \bmod 2)$
5	S_1	S_2	$S_0 \oplus G(N^5 \bmod 2)$	$S_0 \oplus S_1 \oplus G(N^5 \bmod 2)$
6	S_2	S_3	$S_3 \oplus S_0 \oplus G(N^6 \bmod 2)$	$S_3 \oplus S_2 \oplus S_0 \oplus G(N^6 \bmod 2)$
7	S_3	S_0	$S_3 \oplus G(N^7 \bmod 2)$	$G(N^7 \bmod 2)$

The number of iterations of the update function, denoted F, during the key initialization algorithm has no bearing on the success of our attack, just linearly in the number of outputs contained within our map. We have not made a precise investigation of the effect of the constant E upon the attack, although as it is a known constant, it can be factored into the map, with a corresponding increase in complexity.

From observation 8, when the key initialization algorithm is used with $E = 0$ and a 128-bit key, the MAG keystream generation produces only thirty-two possible output bytes. Therefore, any keystream which contains more than 32 distinct output bytes is not generated by MAG with $E = 0$ and a 128-bit key.

Specific indices also show output bytes that are not keystream dependent. For example, we know that $G(N^t \bmod 2)$ has the value of either 0 or $-1 \bmod 2^{32}$. In some cases, the map predicts these output values (for example, in Table 2 when $t = 7$). If the keystream at this point does not contain either of these values, then it is not generated by MAG with $E = 0$ and a 128-bit key.

In other cases, the map predicts output of a single subkey or simple combinations of a few subkeys. For example, in Table 2, when $t = 2$, the map predicts an output value derived solely from subkey S_1, and at $t = 4$, it predicts an output value derived solely from S_2. Although the values of the subkeys are not known, the map allows consistency checks on the obtained keystream. If the checks are satisfied, then with a good measure of confidence, the keystream is generated by MAG under the aforementioned conditions. The chance that a byte produced by a random generator will match the corresponding map entry at index $t + 128F$ is $\frac{1}{128}$. The chance that a keystream byte produced by MAG will match the corresponding entries is 100%. After comparison of a few known keystream bytes against the map, a keystream generated by MAG can easily be distinguished from a random keystream.

This distinguishing attack involves pre-computation with complexity of $\approx F$ operations, where F is the number of iterations of the update function in the key initialization algorithm. Note that this pre-computation is performed once for a given key length, and applies irrespective of the key content. That is, rekeying does not necessitate any additional pre-computation.

Once the pre-computation has been performed, the complexity of the attack is negligible in terms of keystream requirements, memory, and time as it only involves comparing intercepted keystream words with those values predicted by the map. For small values of F, and $E = 0$, both the pre-computation and distinguishing attack can be carried out by hand.

4.2 A Key Recovery Attack on MAG

Observation 5 shows that one quarter of the internal state is leaked through the output function, which merely masks out the highest 24-bits of R_{126}. The map used to distinguish MAG from a random keystream sequence specifies the linear combination of subkeys used to form the output byte, independently of the carry variable C.

Given precomputation of the map for a specific key length, recovery of at least one quarter of the key is trivial by solving some of these equations. For a 128-bit key, about thirty-two bytes of keystream are necessary to recover 32 bits of the key. As noted in [1], information is leaked about the higher bits of some register in the internal state by observing whether $G(N^t \bmod 2)$ is 0 or $-1 \bmod 2^{32}$. However, given that the proposed modifications of MAG examined in the next section permit a complete key recovery, we have not attempted to use this extra information for complete key recovery.

5 Addressing the MAG Modifications

To avoid the distinguishing attack outlined in [1], Vuckovac proposes a modification to the output function of MAG [4]. Instead of taking the keystream from the least significant byte of R_{126}^{t+1}, he proposes outputting byte ($t \bmod 4$) of R_{126}^{t+1}, where R is partitioned into bytes $b_3 \| b_2 \| b_1 \| b_0$. Thus the distinguishing attack in [1] can no longer be applied to the modified MAG.

However, our distinguisher applies as before. As different byte patterns appear in consecutive positions, we merely separate the keystream into four sub-keystreams, by taking every fourth output byte. That is, one sub-keystream consists of the least significant bytes from R_0, another from the second-to-least significant bytes, and so on. We can apply the distinguisher to any of these streams, or to all of them.

For a distinguisher on the 128-bit or 80-bit keys, which are used by ECRYPT Profile 1 ciphers, we do not need extra keystream, and we do not need to generate a new map. The major effect of the modification to the MAG output function is in relation to key recovery. Taking output from all bytes, rather than from just the least significant byte, exposes all of the key bits, and so permits complete

key recovery. We can apply the key recovery attack, outlined in Section 4.2 of this paper, on each stream, and combine the mutually disjoint key information gained from each stream to recover the entire 128- or 80-bit with minimal effort, using around 32 bytes of keystream.

6 Discussion and Summary

In this paper we have shown that the branching operation used in the MAG cipher does not exponentially increase the cipher's complexity, or protect it from standard methods of cryptanalysis, as claimed by MAG's designer in [3]. Instead, we have demonstrated that MAG is a very insecure cipher requiring negligible computing power to attack. We have shown a separate distinguishing attack to that described by [1], and extended our attack to incorporate partial key recovery. More importantly, we have shown that the modification in the output function, intended by Vuckovac to prevent the distinguishing attack of [1], does not prevent the distinguishing attack presented in this paper and actually makes the cipher more vulnerable to key recovery attacks, permitting complete key recovery.

Our attacks are performed under the assumptions that IVs are not used, and that $E = 0$ during key initialization, as it is during keystream generation. This is consistent with the original textual specification of MAG. However, the attacks can still be applied when IVs are used as outlined in [5], by considering the concatenation of key and IV as a key of increased length, of which a portion is known. We believe it feasible, although more complex, to perform the same attacks when E is assigned an arbitrary constant. This is because the known constant is used in a regular way by the key initialization algorithm, such that it does not depend on secret material. The attacker knows the value of E and can factor this into the pre-computation. In any case, we believe that the weakly non-linear addition associated with an arbitrary but known constant adds little security to the cipher.

Problems with MAG start with the absence of a sensible key initialization algorithm. This includes failure to specify an upper limit on the key size, so that for larger key sizes, the cipher becomes vulnerable to standard time-memory-tradeoff attacks. Additionally, because the diffusion at each stage is limited to four stages of the 127-stage register, a guess-and-determine attack becomes possible for large keys (of around 256-bits or larger), as outlined in observation 3.

Irrespective of key size, MAG shows serious weaknesses in that it contains no highly nonlinear components, and directly reveals one-quarter of its internal state through its very weak output function. Within the update function, there is no diffusion between bits within each register stage.

There is an element of security by obscurity in that the specification of MAG is poorly written and at odds with the submitted reference code. We do not believe that MAG can be used as a secure cipher, either as originally described in [3], or with the modifications proposed in [4] and [5]. The modifications required to address the inherent problems in MAG will change its structure so much as to make it unrecognizable.

References

1. Simon Künzli and Willi Meier, *Distinguishing Attack on MAG*, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/053, 2005. http://www.ecrypt.eu.org/stream/mag.html
2. William Millan, *Which software ciphers will survive?*, eSTREAM, ECRYPT Stream Cipher Project, Phorum thread, November 22, 2005. http://www.ecrypt.eu.org/stream/phorum/read.php?1,313
3. Rade Vuckovac. *MAG: My Array Generator (a new strategy for random number generation)*, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/014, 2005. http://www.ecrypt.eu.org/stream/mag.html
4. Rade Vuckovac. *MAG alternating methods notes*, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/068, 2005. http://www.ecrypt.eu.org/stream/mag.html
5. Rade Vuckovac. *MAG Cipher Design Notes*, eSTREAM, ECRYPT Stream Cipher Project, Report 2006/001, 2006. http://www.ecrypt.eu.org/stream/mag.html
6. Stephen Wolfram. *Cryptography with Cellular Automata*, Proceedings of Crypto '85, LNCS 218, Springer-Verlag, 1986, pages 429–432.

On Exact Algebraic [Non-]Immunity of S-Boxes Based on Power Functions*

Nicolas T. Courtois[1], Blandine Debraize[1,2], and Eric Garrido[3]

[1] Axalto Cryptographic Research & Advanced Security,
36-38 rue de la Princesse, BP 45, F-78430 Louveciennes Cedex, France
[2] Versailles University, 45 avenue des États-Unis, 78035 Versailles Cedex France
[3] Thales Communications, 160 bd de Valmy, BP 82, 92704 Colombes Cedex France

Abstract. In this paper we are interested in algebraic immunity of several well known highly-nonlinear vectorial Boolean functions (or S-boxes), designed for block and stream ciphers. Unfortunately, ciphers that use such S-boxes may still be vulnerable to so called "algebraic attacks" proposed recently by Courtois, Pieprzyk, Meier, Armknecht, *et al.* These attacks are not always feasible in practice but are in general very powerful. They become possible, if we regard the S-boxes, no longer as highly-nonlinear functions of their inputs, but rather exhibit (and exploit) much simpler algebraic equations, that involve both input and the output bits. Instead of complex and "explicit" Boolean **functions** we have then simple and "implicit" algebraic **relations** that can be combined to fully describe the secret key of the system.

In this paper we look at the number and the type of relations that do exist for several well known components. We wish to correct or/and complete several inexact results on this topic that were presented at FSE 2004.

We also wish to bring a theoretical contribution. One of the main problems in the area of algebraic attacks is to prove that some systems of equations (derived from some more fundamental equations), are still linearly independent. We give a complete proof that the number of linearly independent equations for the Rijndael S-box (derived from the basic equation $XY = 1$) is indeed as reported by Courtois and Pieprzyk. It seems that nobody has so far proven this fundamental statement.

Keywords: Boolean functions, Power functions, highly non-linear functions S-boxes, design of block and stream ciphers, algebraic attacks, multivariate systems of equations, XL algorithm, Gröbner bases, XSL attack.

Note: This paper is (and should be) dedicated to the memory of Hans Dobbertin [1952-2006], that is recognised for his substantial contributions to the theory of Boolean functions, and did also pioneering work (that remains classified) in the area of algebraic cryptanalysis.

* This work was partially supported by the French Ministry of Research RNRT X-CRYPT project and by the European Commission via ECRYPT network of excellence IST-2002-507932.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 76–86, 2006.
© Springer-Verlag Berlin Heidelberg 2006

1 Introduction

Algebraic attacks are attacks in which a cryptosystem is broken (for example the key, the plaintext, or a signature is computed) by solving a system of multivariate equations over a finite field (e.g. $GF(2)$) that describes the whole cryptosystem. The main idea goes in fact back to Shannon, and the main contributions in the area are (starting from the oldest) [40, 36, 28, 17, 39, 18, 13, 14, 30, 2]. We refer to [21] for a comprehensive survey that outlines respective contributions and (importantly) shows how much in common all these attacks do have.

Algebraic attacks are very successful in cryptanalysis of LFSR-based stream ciphers, see among others [14, 15, 2] as well as for many public key schemes based on multivariate polynomials, for example see [17, 30, 8]. However an essential problem still remains widely open: can an algebraic attack such as XSL or similar break modern block ciphers such as AES faster that the exhaustive search of key space ? - Courtois and Pieprzyk contend it should be possible, see [18, 33, 34], but nobody was so far able to neither prove nor disprove it.

2 How to Measure Algebraic Vulnerability

The notion of algebraic immunity that is used in the literature [4, 21, 1, 5, 6, 7] (and can be defined in several meaningful ways, not only as in [4]) is meant to quantify the security of some cryptosystems (mostly stream ciphers for the notion of [4]) against some algebraic attacks. It does not assure any "immunity", i.e. cannot guarantee security of all ciphers w.r.t. to all algebraic attacks. In this paper we wish to study algebraic immunity in a broader perspective: for Boolean components with several outputs (S-boxes), and for both block and stream ciphers. Our motivation lies in numerous recent proposals of algebraic attacks. A large variety of attacks (cf. among others [21, 1, 14, 15, 2, 16, 18, 20, 12]), and for stream and block ciphers alike, have a commun feature. They all depend on the existence of some "simple" algebraic relations that relate input and output bits of the non-linear components (S-boxes and Boolean functions alike). Thus we will not define a formal notion of algebraic immunity, but will simply look at the critical parameters of the most commonly used algebraic relations of low degree.

In fact, all the S-boxes we study here, are quite weak in this respect, therefore this paper is in fact about algebraic non-immunity or *algebraic vulnerability*. This term also reflects the fact that though some algebraic attacks on ciphers using such weak components are very fast and practical, yet some extremely slow and that may never pose a practical threat (in particular AES has not been shown to be really broken), see [21] for an overview.

Having the aforementioned attacks in mind, the main parameters that do determine algebraic [non-]immunity of an S-box are in general:

1. The size s in bits of the S-box (s stands for **s**ize). In this paper we only consider bijective components $GF(2)^s \rightarrow GF(2)^s$.
2. The type of equations we consider (is usually determined by the kind of monomials we allow, for example quadratic equations).

3. The degree of the monomials that do appear in the equations.
4. The dimension of the space of equations r, (r stands for **r**elations).
5. The sparsity of these equations measured by the number of monomials t (t stands for **t**erms) that do appear in these equations.
6. From (r, s, t) we can compute the number Γ, conjectured by Courtois and Pieprzyk to measure the resistance against the XSL attack. We note that Γ has two different definitions, depending on the version of the XSL attack, see [18]. In this paper we will call Γ the value $\Gamma(r, s, t) = (t/s)^{\lceil t/r \rceil}$ from the `eprint.iacr.org` version of the XSL attack. This version is claimed to be more powerful in practice but requires the internal key scheduling of the block cipher to be built with the same S-box (and otherwise only linear components).
7. Similarly we will call Γ' the definition $\Gamma'(r, s, t) = ((t - r)/s)^{\lceil (t-r)/s \rceil}$ published in the proceeding of Asiacrypt 2002. This version of the XSL attack is more of a theoretical interest: it is simpler to study, does not make any assumption on the key scheduling, but gives in general (but not always) much bigger systems of equations to solve.

In the future, it is possible that a better understanding of the hardness of the problem of solving special systems of multivariate equations will force us to enrich and maybe re-define our notions of algebraic [non-]immunity. The easiest cases may be not the ones that we think, and we may even use totally different types of equations, see for example Section 6.2. in [19]. Nevertheless the types of equations and their main parameters r, s, t that we study in this paper will remain important to look at when studying algebraic attacks on block and stream ciphers.

3 S-Boxes Based on Power Functions over a Finite Field

In this paper we look at various types of exponent functions $X \mapsto X^\alpha$ in a finite field $GF(2^s)$ (we restrict ourselves to the characteristic 2). These functions can be classified according to the exponent α and some exponents are recommended for usage in ciphers on the criteria of satisfying (to some degree) the following two requirements:

1. It is better to use bijective S-boxes (though it is not an obligation for Feistel ciphers). $X \mapsto X^\alpha$ is bijective when $\gcd(\alpha, 2^s - 1) = 1$. For example when $\alpha = 3$ it is known that the function $X \mapsto X^\alpha$ is bijective if and only if s is odd.
2. The exponent function should be non-linear, which excludes all α being a power of 2. Non-linearity is not sufficient, and exponents should rather be very highly non-linear, but maybe not optimal in this respect, as we explain below.

3.1 High Nonlinearity Versus Algebraic Immunity

Non-linearity can be defined in many meaningful ways, depending on the metrics with respect to which we wish the cryptographic components to be "far apart"

from linear components. Highly non-linear components have been widely studied in the literature.

In particular, for power S-boxes, several classes of special exponents have been studied: Gold, Kasami, Dobbertin, Welch, Niho and Inverse, see [9, 3]. These exponents are known to have a very good, optimal or very close to optimal resistance against differential and linear cryptanalysis that is formalised respectively by the notions of Almost-Perfect Nonlinear functions (APN), and [maximally] non-linear functions. We refer to [9, 3] for a bibliography on this topic.

Unfortunately, it turns out that all these "very good" exponents and many other known highly-nonlinear components, are frequently somewhat "very bad" in terms of algebraic immunity, cf. [9, 14, 15, 18, 21, 20]. Yet, research on algebraic attacks and resulting algebraic immunity does not invalidate the previously studied "non-linearity criteria" (such as being an APN) that have been defined for S-boxes and Boolean functions. It rather does complement them, as already suggested in [21, 14, 5]. The new "algebraic relation-related" non-linearity notion, is expected to be related to the other notions and though using (sufficiently large) random S-boxes should be a good idea to avoid all algebraic attacks one can think of, it is also possible to exhibit special components that are reasonably highly non-linear, and at the same time immune to algebraic attacks. For Boolean functions (one output) such constructions have already been studied by Carlet [5] and by Dalai, Gupta and Maitra in [6, 7].

3.2 Prior Work: The FSE 2004 Paper

This paper is meant to be a follow-up to the paper published by Cheon and Lee at FSE 2004 [9]. In this paper the authors follow what can be called "Patarin or/and Courtois-Pieprzyk-style" method (see [36, 18]) for deriving the existence of algebraic equations for the power S-boxes. They do it for 5 other S-boxes known from the literature, find some equations and present 6 theorems to the effect that, some specified equations exist and are linearly independent. In fact as we will see later, 2 of these 6 theorems are not exact, all the other being incomplete (i.e. they do not always take into account all existing equations).

4 Inverse Exponents

The AES so called Inverse S-box, and in fact the power $X \mapsto X^{2^s-2}$ is non-linear for $s > 2$. According to Courtois and Pieprzyk [18], it usually gives $3s - 1$ bi-affine equations (and $5s - 1$ quadratic). This S-box is not at all the same thing that the inverse function in a finite field, 0 is mapped onto itself, and this singularity has surprising and non-trivial consequences, see [20]. One of them is that the number of linearly independent equations is one less than the results incorrectly given in [9] - Theorem 1 of [9] is false.

In the appendix of this paper we give (for a first time) a complete proof that for $s > 2$ the number of linearly independent bi-affine equations is **exactly** $3s - 1$, and for $s > 4$ the dimension of the set of fully quadratic equations

is **exactly** $5s - 1$. Our proof uses the powerful Trace Form representation of Boolean functions and reduces a complex problem of existence and independence of multivariate polynomials into a simpler problem with bivariate polynomials over $GF(2^n)$. The result is confirmed by computer simulations below.

Table 1. Predictions and simulations for the number of linearly independent equations and resulting algebraic immunity for the AES-type S-box $X \mapsto X^{2^s-2}$ over $GF(2^s)$

equation type	size $s =$	2	3	4	5	7	8	9	15	16	17
	Rijndael Inv S-box $X \mapsto X^{-1}, 0 \mapsto 0$										
bi-affine equations $t = s(s+2)+1$	r obtained	5	8	11	14	20	23	26	44	47	50
	expected $= 3s-1$		8	11	14	20	23	26	44	47	50
	$\Gamma = (t/s)^{\lceil t/r \rceil}$	$2^{4.3}$	$2^{4.8}$	$2^{7.9}$	$2^{8.5}$	$2^{12.8}$	$2^{13.4}$	$2^{13.9}$	$2^{24.6}$	$2^{29.2}$	$2^{29.8}$
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$	2^{2}	$2^{4.2}$	$2^{7.2}$	$2^{10.7}$	$2^{18.6}$	$2^{22.9}$	$2^{27.3}$	$2^{57.3}$	$2^{62.7}$	$2^{68.2}$
fully quadratic $t = s(2s+1)+1$	r obtained	7	14	21	24	34	39	44	74	79	84
	expected $= 5s-1$				24	34	39	44	74	79	84
	$\Gamma = (t/s)^{\lceil t/r \rceil}$	$2^{5.4}$	$2^{6.1}$	$2^{6.7}$	$2^{10.8}$	2^{16}	$2^{16.7}$	$2^{21.6}$	2^{35}	$2^{35.6}$	$2^{41.4}$
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$	$2^{4.7}$	$2^{7.5}$	$2^{11.6}$	$2^{23.1}$	2^{42}	$2^{52.2}$	$2^{62.8}$	2^{133}	2^{146}	2^{159}

5 Gold Exponents

Gold exponents (cf. [35, 27, 9]) are functions of type $X \mapsto X^{2^k+1}$ with $\gcd(k, s) = 1$. In [9] we read that these functions are APN, which is not quite true in general, for example when $s = 2, k = 1$. The real result is that all permutation polynomials of this type are APN, i.e. when also $\gcd(2^k + 1, 2^s - 1) = 1$, see [35]. As S-boxes these exponents were first studied by Pieprzyk [38] and Nyberg [35]. Permutation Gold powers are also used in the Matsumoto-Imai multivariate public key scheme and the equations we study below, are precisely the equations that Patarin uses to break this cryptosystem, see [36] for details.

From Theorem 2 of [9] we expect that for every Gold exponent $\alpha = 2^k + 1$ we obtain $3s$ quadratic equations for $k \neq 1$ and $5s$ for $k = 1$. Another theorem of FSE 2004 that is false: for $s = 3$, the S-box $X \mapsto X^3$ gives 14 quadratic equations, instead of 15 expected. In all other cases we studied, looking at Theorem 2 of [9] seems to provide a lower bound for the number of equations found, but this bound is frequently not tight.

For example, for $s = 8$, the S-box $X \mapsto X^5$ (that is not bijective) gives 34 quadratic equations, instead of 24 expected (which is not even a multiple of s). We also get more equations than expected from this theorem for many permutation polynomials, for example $X \mapsto X^5$ for $s = 5$ and for $s = 7$ we get respectively 25 and 28 quadratic equations instead of 15 and 21 expected from Theorem 2 of [9]. Moreover we do not see any regularity here: in the fist case we get $5s$, in the second case $4s$ equations. The algebraic behaviour of Gold exponents is much more complex to understand that the authors of [9] have

Table 2. Comparing simulations to expectations on the number of linearly independent equations and resulting algebraic immunity Γ and Γ', for selected permutation Gold polynomials $X \mapsto X^{2^k+1}, \gcd(k,s) = 1, \gcd(2^k+1, 2^s-1) = 1, 1 \leq k \leq s/2$ (for completeness we also indicate between parentheses r obtained when these conditions are not all satisfied, e.g. polynomials that are not permutations)

equation type	size $s=$	2	3	4	5	7	8	9	15	16	17
	Gold-type S-box $X \mapsto X^3, k=1$										
bi-affine equations $t = s(s+2)+1$	r obtained	(6)	8	(10)	10	14	(16)	18	30	(32)	34
	compare to $2s$		6		10	14		18	30		34
	$\Gamma = (t/s)^{\lceil t/r \rceil}$		$2^{4.8}$		$2^{11.4}$	2^{16}		$2^{20.8}$	$2^{36.8}$		$2^{42.5}$
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$		$2^{4.2}$		$2^{14.3}$	$2^{22.7}$		$2^{31.9}$	$2^{62.6}$		$2^{75.7}$
fully quadratic $t = s(2s+1)+1$	r obtained	(7)	14	(21)	25	35	(40)	45	75	(80)	85
	predicted in [9]		15		25	35		45	75		85
	$\Gamma = (t/s)^{\lceil t/r \rceil}$		$2^{6.1}$		$2^{10.8}$	2^{16}		$2^{21.6}$	2^{35}		$2^{41.4}$
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$		$2^{7.5}$		$2^{22.8}$	$2^{41.7}$		$2^{62.7}$	2^{133}		2^{159}
	Gold-type S-box $X \mapsto X^5, k=2$										
bi-affine equations $t = s(s+2)+1$	r obtained	(5)	(8)	(10)	10	7	(8)	9	15	(16)	17
	compare to s				5	7		9	15		17
	$\Gamma = (t/s)^{\lceil t/r \rceil}$				$2^{11.4}$	$2^{31.9}$		$2^{41.7}$	$2^{73.7}$		2^{85}
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$				$2^{14.3}$	$2^{27.2}$		$2^{36.7}$	$2^{68.1}$		$2^{79.3}$
fully quadratic $t = s(2s+1)+1$	r obtained	(7)	(14)	(21)	25	28	(34)	36	60	(64)	68
	predicted in [9]				15	21		27	45		51
	$\Gamma = (t/s)^{\lceil t/r \rceil}$				$2^{10.8}$	2^{20}		2^{26}	2^{45}		$2^{51.7}$
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$				$2^{22.8}$	$2^{46.8}$		$2^{68.2}$	2^{140}		2^{165}
	Gold-type S-box $X \mapsto X^9, k=3$										
bi-affine equations $t = s(s+2)+1$	r obtained	(6)	(9)	(10)	(10)	14	(12)	(6)	(15)	(16)	17
	compare to s					7					17
	$\Gamma = (t/s)^{\lceil t/r \rceil}$					2^{16}					2^{85}
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$					$2^{22.7}$					$2^{79.3}$
fully quadratic $t = s(2s+1)+1$	r obtained	(7)	(15)	(21)	(25)	35	(32)	(42)	(60)	(64)	68
	predicted in [9]					21					51
	$\Gamma = (t/s)^{\lceil t/r \rceil}$					$2^{16.1}$					$2^{51.7}$
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$					$2^{41.7}$					2^{165}
	Gold-type S-box $X \mapsto X^{17}, k=4$										
bi-affine equations $t = s(s+2)+1$	r obtained	(5)	(8)	(14)	(10)	(14)	(36)	18	15	(16)	17
	compare to s							9	15		17
	$\Gamma = (t/s)^{\lceil t/r \rceil}$							$2^{20.8}$	$2^{75.7}$		2^{85}
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$							$2^{31.9}$	$2^{68.1}$		$2^{79.3}$
fully quadratic $t = s(2s+1)+1$	r obtained	(7)	(14)	(26)	(25)	(35)	(70)	45	60	(68)	68
	expected							27	45		51
	$\Gamma = (t/s)^{\lceil t/r \rceil}$							$2^{21.6}$	2^{45}		$2^{51.7}$
	$\Gamma' = \left(\frac{t-r}{s}\right)^{\lceil \frac{t-r}{s} \rceil}$							$2^{62.7}$	2^{140}		2^{165}

expected, and their results on Γ algebraic immunity of S-boxes are only upper bounds.

Observations on Algebraic Immunity: In Table 2 we see that for XSL attacks both types of equations are interesting. For some S-boxes bi-linear equations give a lower Γ, for other S-boxes, better attacks will be obtained with fully quadratic equations. We also observe, as expected, that usually Γ' is much larger than Γ, but in some cases it isn't.

6 Dobbertin Exponents

Dobbertin exponents are following [9, 24] the power functions of the form $X \mapsto X^{2^{4k}+2^{3k}+2^{2k}+2^k-1}$ over $GF(2^s)$ with $s = 5k$. From Theorem 6 of [9] we expect that there should give s quadratic equations. Again there are conter-examples for this: for example $X \mapsto X^{4679}$ over $GF(2^{15})$ is a Dobbertin permutation that gives only 12 quadratic equations instead of $s = 15$ expected. Some other examples can be explained, for example $X \mapsto X^{29}$ is a Dobbertin permutation over $GF(2^5)$ gives $24 = 5s - 1$ quadratic equations, and this is because this case is degenerated, (we thank the anonymous referee for pointing this out) X^{29} lies in the same cyclotomic coset that the inverse exponent 30.

We note that the Theorem 6 of [9] is neither a lower bound nor an upper bound, and it seems that the correct lower bound for Dobbertin exponents would be $4s/5 = 4k$ and not $s = 5k$.

7 Niho Exponents

Niho exponents are defined in [25, 9] as functions of type $X \mapsto X^{2^m+2^{m/2}-1}$ over $GF(2^s)$ with $s = 2m + 1$ and m even, or $X \mapsto X^{2^m+2^{(3m+1)/2}-1}$ over $GF(2^s)$ with $s = 2m + 1$ and m odd. Though from Theorem 5 of [9] we learn that there should be s linearly independent quadratic equations, again we have found that there are more of them.

For example $X \mapsto X^{39}$ is a Niho permutation polynomial over $GF(2^7)$, and it gives as many as $21 = 3s$ quadratic equations, instead of $s = 7$ expected.

We not also that $X \mapsto X^{39}$ is a Niho permutation polynomial over $GF(2^7)$, and it gives $21 = 3s$ quadratic equations, instead of $s = 7$ expected. This however can be explained: it lies in the same cyclotomic class as the inverse of the Kasami exponent 57.

8 Welch Exponents

Welch exponents are following [24, 9] functions of type $X \mapsto X^{2^m+3}$ over $GF(2^s)$ with $s = 2m + 1$. From Theorem 4 of [9] we learn that for these functions there are as many as $9s$ or $10s$ quadratic equations. Unfortunately these are obtained at the cost of introducing additional variables z_i, and we may still compute Γ but

it does not pertain exactly to the XSL attack anymore. However we may deduce from Table 6 that among these there are s equations (and $2s$ when $m = 2$) that do not use these additional variables.

This prediction is not at all confirmed by our simulations. For example $X \mapsto X^5$ is a Welch permutation over $GF(2^3)$, and it gives $14 = 5s - 1$ quadratic equations, instead of $s = 3$ expected. Other examples are $X \mapsto X^7$ over $GF(2^5)$, and $X \mapsto X^{11}$ over $GF(2^7)$, that are Welch permutations, and give respectively $25 = 5s$ and $21 = 3s$ quadratic equations, instead of $2s = 10$ and $s = 7$ we expect.

Only some examples confirm what we expect from Theorem 4 and Table 6 of [9]. For example $X \mapsto X^{19}$ over $GF(2^9)$ and $X \mapsto X^{131}$ over $GF(2^{15})$ are Welch permutations and in both cases there are indeed s quadratic equations.

9 Kasami Exponents

Kasami exponents are defined in [31, 9] as functions of type $X \mapsto X^{2^{2m}-2^m+1}$ over $GF(2^s)$ with $\gcd(m, s) = 1$ and $1 \leq m \leq s/2$. From Theorem 3 of [9] we learn that for these functions there are as many as $7s$ or $10s$ quadratic equations (but again they introduce additional variables). Out of them s quadratic equations do not involve additional variables.

Again, this is not at all confirmed by our simulations. For example $X \mapsto X^{13}$ is a Kasami permutation over $GF(2^7)$, $GF(2^9)$ and $GF(2^{11})$with $k = 2$, and it gives respectively $21 = 3s$, $18 = 2s$ and $22 = 2s$ quadratic equations instead of s expected. Another example is $X \mapsto X^{57}$ which is a Kasami function (not a permutation) over $GF(2^8)$, $GF(2^{10})$ and $GF(2^{14})$with $k = 3$, and in all these cases we get $2s$ quadratic equations, instead of s expected.

10 Which S-Box Has the Lowest Algebraic Immunity?

In this paper we see that in most cases the algebraic behaviour of power functions over finite fields is far from being simple and predictable. For many results of [9] up to 5 times more equations do exist which results in a much lower algebraic immunity than expected.

Which S-box is the worse ? During the Asiacrypt 2002 presentation, Courtois conjectured (based on some early comparisons) that there is no non-linear S-box that allows to write more multivariate relations that for the AES S-box (i.e. in terms of algebraic immunity AES uses the worst S-box that exists). In this paper we show that strictly speaking this conjecture is not true.

We found that if s is odd, the function $X \mapsto X^3$ over $GF(2^s)$ is an APN permutation S-box and gives usually (but not always as claimed in [9], see simulations in Table 2) as many as $5s$ equations instead of $5s - 1$ for the inverse S-box. Yet, in an algebraic attack on AES such as in [18] it is still possible to use $5s$ equations for all S-boxes of AES and this with pretty good probability. Therefore, arguably, an S-box based on $X \mapsto X^3$ is in fact only very slightly worse than the AES S-box. In some sense the Courtois conjecture remains valid.

Remark: For one $s = 4$ (and only for this s) we can even do better than $5s$, and we have $21 = 5s + 1$ quadratic equations for the AES S-box itself on 4 bits. This is due to the following fact proven by Courtois and Pieprzyk in [18]: there is no S-box on 4 bits for which there would be less than 21 equations. From this one can conjecture that, when $s > 4$, there (maybe) is no non-linear S-box that would give strictly more than $5s$ quadratic equations.

11 Conclusion

Algebraic attacks work by creating algebraically dependent but linearly independent sets of equations, derived from some given initial set of equations. The complexity of algebraic attacks on block ciphers does greatly depend on whether there are sufficiently many linearly independent equations compared to some evaluation. In this paper we showed that this problem is complex and not trivial even for tiny systems of equations resulting in a finite field from one single power-function S-box.

At FSE 2004 a paper was published with 6 theorems that determine the number of linearly independent multivariate quadratic equations and resulting algebraic immunity for 6 different highly non-linear power S-boxes known from the literature. In this paper we showed that all these 6 results are false and in some cases heavily underestimate the number of linearly independent algebraic relations (up to 5 times). For Inverse and Dobbertin, the actual dimension may also be lower than claimed.

For the time being, computer simulations rather than extant theory, are the only way known to determine correctly the number of linearly independent equations, even for one single S-box. Nevertheless, we managed to solve the problem completely for the AES S-box: we give a complete proof using the Trace Form of Boolean functions that the number of linearly independent equations is indeed as established by heuristic derivation combined with computer simulations by Courtois and Pieprzyk. It seems that it is the first time such an exact result is proved. Moreover, our proof methodology should be of independent interest and might help to prove the independence of more complex systems of equations that arise in algebraic attacks.

References

1. Frederik Armknecht: *On the Existence of low-degree Equations for Algebraic Attacks,* preprint available at `eprint.iacr.org/2004/185/`. Also presented at SASC Ecrypt workshop (State of the Art in Stream Ciphers), Bruges, Belgium, October 14-15 2004.
2. Frederik Armknecht, Matthias Krause: *Algebraic Atacks on Combiners with Memory,* Crypto 2003, LNCS 2729, pp. 162-176, Springer.
3. Anne Canteaut, Marion Videau: *Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis,* Eurocrypt 2002, LNCS 2332, Springer.

4. Claude Carlet, Will Meier and Enes Pasalic: *Algebraic Attacks and Decomposition of Boolean Functions*, Eurocrypt 2004, pp. 474-491, LNCS 3027, Springer, 2004.
5. Claude Carlet: *Improving the algebraic immunity of resilient and nonlinear functions and constructing bent functions,* preprint available at `eprint.iacr.org/2004/276.pdf`.
6. Deepak Kumar Dalai, Kishan Chand Gupta and Subhamoy Maitra: *Results on algebraic immunity for cryptographically significant Boolean functions,* In Indocrypt 2004, LNCS 3348, pp. 92-106, Springer, 2004.
7. Deepak Kumar Dalai, Kishan Chand Gupta and Subhamoy Maitra: *Cryptographically Significant Boolean functions: Construction and Analysis in terms of Algebraic Immunity,* To appear in FSE 2005, LNCS, Springer.
8. Jiun-Ming Chen, Nicolas Courtois and Bo-Yin Yang: *On Asymptotic Security Estimates in XL and Gröbner Bases-Related Algebraic Cryptanalysis,* ICICS'04, LNCS 3269, pp. 401-413, Springer, 2004.
9. Jung Hee Cheon and Dong Hoon Lee: *Resistance of S-boxes against Algebraic Attacks,* In FSE 2004, Springer. Can be found at `http://www.math.snu.ac.kr/~jhcheon/Published/2004_FSE/FSE04_CL.pdf`.
10. Don Coppersmith, Shmuel Winograd: "Matrix multiplication via arithmetic progressions", J. Symbolic Computation (1990), 9, pp. 251-280.
11. Paul Camion, Claude Carlet, Pascale Charpin and Nicolas Sendrier, *On Correlation-immune Functions,* In Crypto'91, LNCS 576, Springer, pp. 86-100.
12. Nicolas Courtois: *Feistel Schemes and Bi-Linear Cryptanalysis,* in Crypto 2004, LNCS 3152, pp. 23-40, Springer, 2004.
13. Nicolas Courtois: *Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt,* ICISC 2002, LNCS 2587, Springer.
14. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, Warsaw, Poland, LNCS, Springer.
15. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback,* In Crypto 2003, LNCS 2729, pp: 177-194, Springer 2003.
16. Nicolas Courtois: *Algebraic Attacks on Combiners with Memory and Several Outputs,* ICISC 2004, LNCS, to appear in Springer in early 2005. Extended version available on `http://eprint.iacr.org/2003/125/`.
17. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*, Cryptographers' Track Rsa Conference 2001, LNCS 2020, Springer, pp. 266-281.
18. Nicolas Courtois and Josef Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Asiacrypt 2002, LNCS 2501, Springer, a preprint with a different version of the attack is available at `http://eprint.iacr.org/2002/044/`.
19. Nicolas Courtois, Guilhem Castagnos and Louis Goubin: *What do DES S-boxes Say to Each Other ?* Available on `eprint.iacr.org/2003/184/`.
20. Nicolas Courtois: *The Inverse S-box, Non-linear Polynomial Relations and Cryptanalysis of Block Ciphers,* in AES 4 Conference, Bonn May 10-12 2004, LNCS 3373, pp. 170-188, Springer.
21. Nicolas Courtois: *General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers,* in AES 4 Conference, Bonn May 10-12 2004, LNCS 3373, pp. 67-83, Springer.
22. Joan Daemen, Vincent Rijmen: *AES proposal: Rijndael,* The latest revised version of the proposal is available on the Internet, `http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf`
23. Hans Dobbertin: *One-to-One Highly Nonlinear Power Functions on GF(2^n)*, Appl. Algebra Eng. Commun. Comput. 9(2): 139-152 (1998).

24. Hans Dobbertin: *Almost perfect nonlinear power functions on GF(2^n): the Welch case.* IEEE Transactions on Information Theory, 45(4):1271-1275, 1999.
25. Hans Dobbertin: *Almost perfect nonlinear power functions on GF(2^n): the Niho case.* Information and Computation, 151:57–72, 1998.
26. Jovan Dj. Golic: *On the Security of Nonlinear Filter Generators,* FSE'96, LNCS 1039, Springer, pp. 173-188.
27. R. Gold: *Maximal recursive sequences with 3-valued recursive crosscorrelation functions,* IEEE Transactions on Information Theory, 14:154–156, 1968.
28. Thomas Jakobsen: *Cryptanalysis of Block Ciphers with Probabilistic Non-Linear Relations of Low Degree,* Crypto 98, LNCS 1462, Springer, pp. 212-222, 1998.
29. Thomas Jakobsen, Lars R. Knudsen: *The Interpolation Attack on Block Ciphers,* FSE 97, LNCS 1267, Springer, pp.28-40, 1997.
30. Antoine Joux, Jean-Charles Faugère: *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases*, Crypto 2003, LNCS 2729, pp. 44-60, Springer.
31. T. Kasami: *The weight enumerators for several classes of subcodes of the second order binary Reed-Muller codes.* Information and Control, 18:369–394, 1971.
32. Will Meier, Enes Pasalic and Claude Carlet: *Algebraic Attacks and Decomposition of Boolean Functions,* In Eurocrypt 2004, pp. 474-491, LNCS 3027, Springer, 2004.
33. Sean Murphy, Matt Robshaw: *Essential Algebraic Structure within the AES,* Crypto 2002, LNCS 2442, Springer.
34. Sean Murphy, Matt Robshaw: *An analysis of the XSL attack and it's impact on the security of AES,* Nessie report, `https://www.cosic.esat.kuleuven.ac.be/nessie/reports/phase2/Xslbes8_Ness.pdf`.
35. Kaisa Nyberg and Lars R. Knudsen: *Provable security against differential cryptanalysis,* Journal of Cryptology, vol 8, n. 1, 1995, pp. 27-37, also appears in Crypto'92, LNCS 746, pp. 566-574, Springer, 1992.
36. Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88,* Crypto'95, Springer, LNCS 963, pp. 248-261, 1995.
37. Jacques Patarin: *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms,* in Eurocrypt'96, Springer, pp. 33-48. The extended version can be found at `http://www.minrank.org/hfe.ps`
38. J. Pieprzyk: *On bent premutations,* Technical Report CS 91/11; The University of New South Wales, Australia.
39. Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
40. Claude Elwood Shannon: *Communication theory of secrecy systems,* Bell System Technical Journal 28 (1949), see in patricular page 704.
41. Volker Strassen: *Gaussian Elimination is Not Optimal,* Numerische Mathematik, vol 13, pp 354-356, 1969.
42. A.M. Youssef and G. Gong: *Hyper-bent Functions,* In Eurocrypt 2001, LNCS 2045, pp 406-419, Springer, 2001.

A Proof of the Main Theorem

Due to the space limitations of these proceedings, the proof can be found in the full version of this paper available from the authors or at `eprint.iacr.org/2005/203`.

Augmented Certificate Revocation Lists

A. Lakshminarayanan and T.L. Lim

Institute for Infocomm Research, Singapore
{lux, tllim}@i2r.a-star.edu.sg

Abstract. We present a simple yet clever extension to the delta certificate revocation list(CRL) [1], the augmented certificate revocation list (ACRL). ACRLs contain revocation updates only and certificate verifiers construct complete CRLs locally. Locally constructed complete CRLs are identical to complete CRLs issued by the CRL issuer. So certificate verifiers need not download complete CRLs. ACRLs are much smaller in size compared to complete CRLs providing significant network savings. Contrary to existing opinion - that CRLs cannot provide efficient online certificate status - we present an ACRL based online certificate status scheme which has many advantages over OCSP [2]. ACRLs are backward compatible and can easily be integrated into existing X.509 CRL based schemes.

1 Introduction

Many security protocols use digitally signed public-key certificates. A public-key certificate binds a public key with the identity of the owner of the corresponding private key. X.509 is an industry standard that defines the format of public-key certificates [1]. The primary contents of a public-key certificate include a public key, the certificate's unique identifier e.g. serial number, the certificate owner's identity, the certificate expiry date and certificate extensions (in this paper, public key certificate means X.509 version 3 certificate). To ensure the authenticity of the certificate owner's identity, certificates are digitally signed by a trusted third party called certificate authority (CA). The binding between the public key and the owner identity can be broken if the associated private key is compromised e.g. stolen or lost. In such circumstances, the certificate needs to be revoked and this revocation information made available to certificate verifiers. Providing timely and reliable certificate status information is expensive. Even though certificate status schemes have been proposed in the past, the most popular schemes are periodically issued certificate revocation lists (CRLs) [1] and IETF's online certificate status protocol (OCSP) [2].

To compare certificate status schemes, we use the following performance metrics 1) Network costs: bandwidth, total certificate status data downloaded and peak request rates 2) Computational costs and 3) Operational costs: server security and replication. Section 2 discusses popular CRL schemes. In section 3, we present our augmented certificate revocation list (ACRL), a simple yet clever extension to the delta CRL. ACRLs contain revocation updates only and certificate verifiers (henceforth referred to as clients) construct complete CRLs locally.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 87–98, 2006.
© Springer-Verlag Berlin Heidelberg 2006

These locally constructed complete CRLs are identical to complete CRLs issued by the CRL issuer. So clients do not need to download complete CRLs. Using our PKI model, we show that ACRLs are significantly superior to any existing CRL scheme. We also present our delta ACRL scheme and show that it can provide efficient online certificate status. In section 4, we show that our ACRL scheme has significant advantages over OCSP. ACRLs are delta CRLs with X.509 CRL extensions - so using ACRLs, existing CRL based schemes can be easily upgraded to provide online certificate status.

2 Certificate Revocation Lists

A complete CRL is a time-stamped digitally signed list of serial numbers (or other certificate identifiers) of un-expired certificates that have been revoked by the revocation authority (henceforth referred to as the CRL issuer). CRLs (in this paper, CRL means X.509 version 2 CRL) are updated and issued at regular intervals even if the revocation list has not changed. CRLs can be distributed using a variety of protocols, e.g. LDAP, HTTP and FTP [1]. Revoked certificates remain in the CRL list until they expire. The key elements of a CRL is shown in Fig.1.

CRL Header (approx. 50 bytes)
- Issuer's name: 32 bytes (if X.500 name used)
- Date/time of CRL issuance (`thisUpdate`): 6 bytes
- Date/time of next CRL issuance (`nextUpdate`): 6 bytes

List of revoked certificates (9 bytes per revoked certificate)
- Serial number : 3 bytes
- Revocation date : 6 bytes
- CRL entry extensions (e.g. revocation reason)

CRL general extensions (e.g. CRL Number)
Signature of CRL issuer (RSA 1024 bit signature, approx. 130 bytes)

Fig. 1. Key elements of X.509 v2 CRL

CRLs are digitally signed, so can easily be replicated and shared. Expensive trusted servers are not necessary to distribute CRLs. Since CRLs are not generated on a per-request basis, only a few fixed number of CRLs need to be created in any period of time making CRL generation computationally inexpensive. These reasons make CRLs attractive. A trivial certificate status scheme involves publishing complete CRLs at regular periodic intervals. However, when PKI population is large, complete CRLs tend to become large imposing high network costs on the CRL distribution server. Moreover, if clients have limited bandwidth capability e.g. dial-up connections, long download times can be annoying. Factors influencing complete CRL size include PKI population, revocation rate and certificate validity period, the first two factors usually beyond

the CRL issuer's control. Since expired certificates can be removed from a CRL, it is tempting to set shorter certificate validity periods to reduce complete CRL size. But this is not preferable since frequent certificate generation and updates leads to greater user inconveniences and higher administration costs. To alleviate high CRL distribution costs - especially for large PKIs, several CRL variants have been proposed.

Segmented CRLs. It is possible to arbitrarily divide the certificate population into many partitions, each partition being associated with a CRL distribution point [1]. While segmenting CRLs does not reduce peak CRL request rate, it reduces the size of each CRL downloaded reducing peak bandwidth [3]. However, if relying clients operate off-line, segmentation is not useful. Relying parties that operate off-line will not know which certificates they will be validating at the time they obtain CRLs and so need to download all segmented CRLs [3]. Even if clients are on-line - in the worst-case - clients need to download CRLs from all distribution points.

Over-Issued CRLs. Traditionally, a new CRL is not issued (and clients with unexpired CRLs do not request new CRLs) until the time specified in the `nextUpdate` field of the currently issued CRL has been reached. However, if CRLs are over-issued, i.e. a new CRL is issued before the previous one expires (before the `nextUpdate` time of the previous CRL has been reached), some relying clients will retrieve this new CRL while others continue to use previously issued yet valid CRL. Though Cooper showed in [3] and [4] that over-issued CRLs are effective in reducing peak bandwidth, using over-issued CRLs results in different clients possessing different revocation lists at the same time.

Delta CRLs. A delta CRL is a time-stamped digitally signed list containing information of new certificate revocations that occurred since the issuance of a prior base CRL. A base CRL is a complete CRL to which the revocation list in the delta CRL needs to be applied to produce the latest list of revoked certificates. A delta CRL uses its `DeltaCRLIndicator` extension [1] to identify the base CRL it refers to, the base CRL identified by its `CRLNumber`. If clients possess secure repositories, it is possible to completely eliminate the need to download base CRLs because such clients can keep updating their revoked certificate lists using just delta CRLs. This approach is not practical because secure repositories - which also need to be maintained for long time periods - are expensive especially on simple clients such as desktops and mobile clients.

Sliding Window Delta CRLs. Cooper [4] showed that if delta CRLs are issued in the traditional manner, i.e. base CRLs issued at regular intervals and delta CRLs issued regularly but more frequently, the performance gain of using delta CRLs is not as significant as one would expect. Cooper's sliding window delta CRL scheme [4] uses a large window size (time interval between delta CRL issuance and the referenced base CRL issuance) for all delta CRLs which reduces peak bandwidth loads considerably. Clients using the sliding window CRL scheme use local repositories to update their revocation lists. These

local repositories not only should be secure but also cannot be shared with other relying clients.

In this paper, we use a model almost identical to Cooper's PKI model [3] to analyze the performance of various certificate status schemes. This model is characteristic of large PKI populations. As in [3], we make the following assumptions. Clients cache CRLs and do not download a new CRL until the cached CRL's `nextUpdate`. For analysis purposes, we assume just one CRL distribution server. Once a certificate is revoked, its status is reflected in the CRL until the certificate expires. A revoked certificate will remain in a CRL for approximately half its lifetime [3].

- $N = 300{,}000$ (PKI population size)
- $P = 365$ days (certificate validity period)
- $v = 10$ certificates validated per day by each client
- $r_{revoke} = 200$ revocations per day
- $r_{expiry} = 200$ revoked certificates expire per day
- At steady state, $r_{revoke} = r_{expiry}$. In [3] and [4], Cooper makes the same assumption.

Assuming that revoked certificates remain in a CRL for half their lifetime, the size of a CRL (S_{CRL}) is given by (1), where S_H is the CRL header size (180 bytes: 50 bytes CRL header info + 130 bytes RSA 1024 bit signature) and S_E the CRL entry size (9 bytes). If T is the base CRL issuance interval and M delta CRLs are issued every T interval, the size of the j^{th} delta CRL (S_{DCRL}) is given by (2). If client validation requests arrive independent of each other, an exponential inter-arrival probability density function can be used to derive the request rate (R) for downloaded CRLs as given by (3) [3]. The total bandwidth is a product of CRL size and request rate (4). CRL data downloaded ($CRLData_T$) during one CRL update interval T is given by (5). Table 1 shows the performance of different CRL schemes under our PKI model.

$$S_{CRL} = S_H + 0.5 S_E r_{revoke} P \ . \tag{1}$$

$$S_{DCRL}(j,T) = S_H + S_E(jT/M) r_{revoke} \ , \ j = 1 \ to \ M \ . \tag{2}$$

$$R(t) = Nve^{-vt} \ . \tag{3}$$

$$T_{BW}(t) = S_{CRL} R(t) \ . \tag{4}$$

$$CRLData_T(T) = \int_0^T S_{CRL} R(t) dt = S_{CRL} N(1 - e^{-vT}) \ . \tag{5}$$

3 Augmented Certificate Revocation Lists (ACRLs)

The attributes of a periodically issued X.509 complete CRL that change with every fresh CRL issuance are 1) date and time of CRL issuance - `thisUpdate` 2) date and time of next CRL issuance - `nextUpdate` 3) CRL number 4) latest list

Table 1. Performance of different CRL schemes

CRL scheme	CRL size (KB)	Peak request rate (/sec)	Peak bandwidth (KB/sec)	Total data transferred (GB/day)
Base CRL only 12 hrs interval	321	34.7	11138	182.4
Segmented CRL 10 segments 12 hrs interval	32.1	34.7	1113.8	182.4 (worst case) 18.2 (best case)
Over-issued CRL 12 hrs interval, over-issued every 4 hrs	321	11.1	3562	182.4
Traditional delta CRL base interval: 12 hrs delta interval: 1 hr	base CRL: 321 delta CRL: 0.25 to 1.05	base CRL: 22.9 delta CRL: 34.7	7356	121.8
Sliding window delta CRL base interval: 12 hrs delta interval: 1 hr window size: 18 hrs	base CRL: 321 delta CRL: 1.5	base CRL: 0.019 delta CRL: 34.7	58.1	3.9
Sliding window delta CRL base interval: 12 hrs delta interval: 1 min window size: 18 hrs	base CRL: 321 delta CRL: 1.5	base CRL: 0.019 delta CRL: 34.7	58.1	4.8

of revoked certificates and 5) signature over CRL contents. Changes in the latest list of revoked certificates include new revocations reported between issuance of previous complete CRL and the freshly issued complete CRL as well as deletions of revoked expired certificates during the same time. This list can be ordered (e.g. certificate serial numbers in ascending order) using the ordered list CRL extension [1]. To construct ACRLs, we follow certain rules.

- The list of revoked certificates in a CRL is ordered
- CRL issuer name and CRL update intervals are publicly known attributes
- When a delta CRL is issued, a complete CRL is also issued at the same time. This complete CRL can serve as a base CRL for delta CRLs issued later
- All CRLs have CRL numbers. The `CRLNumber` (and `thisUpdate` field) of the base CRL and delta CRL issued at the same time is same (since CRL update intervals are publicly known, the `nextUpdate` attribute of CRLs can be easily derived)
- CRLs expire when the CRL issuer's signing key expires

An augmented CRL (ACRL) is a delta CRL that carries the CRL issuer's digital signature over the base CRL (base CRL issued at the same time as the ACRL) in an additional CRL general extension (this CRL general extension henceforth referred to as signature bytes extension). Fig.2 shows the ACRL issuance time-line. The data structures of X.509 Base and ACRLs are very similar, so a client with the latest ACRL (and the previous base CRL this ACRL refers to) possesses all key attributes of the latest base CRL. The client obtains the `CRLNumber`, `thisUpdate` and `nextUpdate` fields from the latest ACRL. The revoked certificates list contained in the latest base CRL is a union of two lists - the list contained in the latest ACRL and the revocation list of the previous base CRL (which the latest ACRL references). Since an ACRL carries the signature bytes of the base CRL (issued at the same time) in its signature bytes

CRL general extension, a client can construct locally the latest base CRL using just the latest ACRL and the previous base CRL. Similarly, if the client does not possess the previous base CRL (or any prior base CRL), it can download the previous ACRL (or prior ACRLs) to construct the previous base CRL (or any prior base CRL). Hence base CRLs can be constructed with just ACRLs without any need to download large base CRLs (a client needs to download one base CRL though, the very first time it issues a CRL request. Thereafter, it can locally construct base CRLs using just ACRLs).

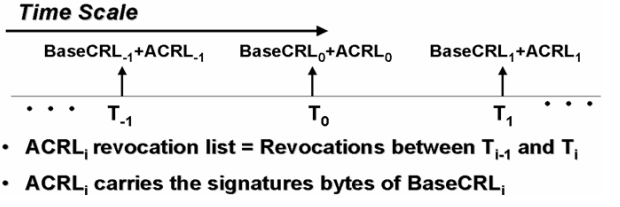


Fig. 2. ACRL issuance timeline

3.1 Handling Expired Certificates

A CRL revocation list contains the serial numbers of un-expired revoked certificates. In earlier CRL schemes, the CRL issuer regularly deletes expired certificate entries from base CRLs. With ACRLs, how do clients know which revoked certificates have expired? We use a X.509 CRL general extension (henceforth referred to as expired revoked certificate extension) to remove expired revoked certificates. This extension contains the serial numbers of revoked certificates that have expired since the issuance of the previous ACRL. Assuming that the certificate serial number is 3 bytes, the size of an ACRL is given by (6) where S_{AH} is the ACRL's header size, S_E size of a CRL entry, S_{AE} size of a ACRL entry and T the ACRL issuance interval (In our model $r_{revoke} = r_{expiry}$, so $S_{AE} = S_E + 3$. S_{AH} is 310 (= 180 + 130) bytes, approx. 130 bytes for RSA 1024 bit signature).

3.2 Constructing Base CRLs Locally

Let T_0 be current time and $BaseCRL_j$ and $ACRL_j$ be the j^{th} base CRL and ACRL respectively, both issued at time T_j, T being the ACRL update interval. $BaseCRL_0$ and $ACRL_0$ are the latest base CRL and ACRL respectively, both issued at time T_0. To construct $BaseCRL_0$ with $ACRL_0$, a client needs $BaseCRL_{-1}$. If the client doesn't possess $BaseCRL_{-1}$ but possesses $BaseCRL_{-2}$, the client downloads $ACRL_{-1}$, constructs $BaseCRL_{-1}$'s revocation list and then $BaseCRL_0$. If a client possesses a base CRL ($BaseCRL_{-i}$) issued i ACRL issuance time intervals earlier, the client needs all ACRLs - $ACRL_{-i+1}$ to $ACRL_0$ (which we shall refer to as intermediate ACRLs) to construct $BaseCRL_0$. When a client constructs the latest base CRL locally, the

latest base CRL's signature bytes is sufficient to guarantee the integrity of the locally constructed base CRL. The client does not need the CRL issuer's digital signature on each intermediate ACRL. A client just needs to download the latest base CRL's signature bytes (`BaseCRLSB`) and the list of revoked certificates (and revoked expired certificates) of each intermediate ACRL to construct the latest base CRL. The data structure that contains just the list of revoked certificates (and the list of revoked expired certificates) of an ACRL is called the augmented CRL `RevocationInfo` (ACRLRI). The size of an ACRLRI is given by (7). If a client suspects that any ACRL `RevocationInfo` has been altered, it can request ACRLs in their entirety and verify the CRL Issuer's digital signature over each such ACRL. The base CRL signature byte size (S_{SB}) is approximately 130 bytes (RSA 1024 bit signature).

In our scheme, the CRL number (`CRLNumber`) is a function of the time interval the ACRL is issued. If t is time and T the ACRL update issuance interval, the CRLNumber is given by (8) where K and C are publicly known constants. The CRL issuer constructs a fresh ACRL at the start of every ACRL update interval and uploads it to the ACRL distribution server. This server extracts the associated ACRL `RevocationInfo` and base CRL's signature bytes from the ACRL and makes them available to clients.

$$S_{ACRL}(T) = S_{AH} + (S_E r_{revoke} + 3r_{expiry})T = S_{AH} + S_{AE}\ r_{revoke} T\ . \tag{6}$$

$$S_{ACRLRI}(T) = S_{AE}\ r_{revoke} T\ . \tag{7}$$

$$CRLNumber(t, T) = K\ \lfloor t/T \rfloor + C\ . \tag{8}$$

3.3 Performance Analysis

In the time interval between T_0 and T_1, any client requesting certificate status will need $BaseCRLSB_0$ and $ACRLRI_0$. Assuming that CRL requests arrive independent of each other, the request rate (RA_0) for these two data structures is given by (9). The same clients if they do not possess the revocation information present in $ACRL_{-1}$ (or $ACRL_{-i}$), will request for $ACRLRI_{-1}$ (or $ACRLRI_{-i}$). The request rate for the ACRL `RevocationInfo` issued i ACRL intervals earlier is given by (10). Assuming that the first ACRL was issued q intervals earlier and at steady-state q is a very large number, the combined request rate and peak combined request rate for ACRL data structures is given by (11) and (12) respectively. The total bandwidth (T_{BW}) is a summation of the bandwidth contributions from all downloaded ACRL data structures (13), and the peak bandwidth (P_{BW}) is given by (14). The total data downloaded from the CRL distribution server in one ACRL update interval T is given in (15). As we reduce the ACRL update interval to shorter values, the peak bandwidth and total ACRL data downloaded in one day approach values given by (16) and (17) respectively.

$$RA_0(t, T) = Nve^{-vt}\ (0 \le t < T)\ . \tag{9}$$

$$RA_{-i}(t, T) = Nve^{-v(t+iT)}\ (0 \le t < T)\ . \tag{10}$$

$$\text{Combined Request Rate, } RA(t) = \lim_{q\to\infty} \sum_{i=0}^{q-1} RA_{-i}(t,T)$$
$$= Nve^{-vt}/(1-e^{-vT}) \; . \tag{11}$$

$$\text{Peak Combined Request Rate, } P_{RA}(T) = Nv/(1-e^{-vT}) \; . \tag{12}$$

$$T_{BW}(t,T) = \lim_{q\to\infty} \sum_{i=0}^{q-1} (\text{Size of } ACRLRI_{-i} \times RA_{-i}(t,T)) + S_{SB}RA_0(t,T)$$
$$= Nve^{-vt}(S_{SB} + S_{AE}r_{revoke}T/(1-e^{-vT})) \; . \tag{13}$$

$$P_{BW}(T) = Nv(S_{SB} + S_{AE}r_{revoke}T/(1-e^{-vT})) \; . \tag{14}$$

$$ACRLData_T(T) = \lim_{q\to\infty} \sum_{i=0}^{q-1} (\text{No. of Reqs. for } ACRLRI_{-i} \times \text{Size of } ACRLRI_{-i})$$
$$= N\left(S_{SB}(1-e^{-vT}) + S_{AE}r_{revoke}T\right) \; . \tag{15}$$

$$P_{BW}(T \to 0) = N(S_{SB}v + S_{AE}r_{revoke}) \; . \tag{16}$$

$$ACRLData_{24Hours}(T \to 0) = 86400N(S_{SB}v + S_{AE}r_{revoke}) \; . \tag{17}$$

Table 2 shows the performance of the ACRL scheme for different ACRL update intervals for our PKI model. Comparing Tables 1 and 2, it is clear that the ACRL scheme performs significantly better than any other existing CRL scheme including Cooper's sliding window scheme [4], with respect to peak bandwidth and total CRL data downloaded. Complete CRLs provide a full list of all revoked certificates every time they are downloaded. This is highly inefficient since clients download revocation information they already possess. Using ACRLs, which contain only revocation updates, clients do not need to download complete CRLs making ACRL performance significantly better.

3.4 Delta ACRLs

Even though the peak bandwidth reduces as we reduce the ACRL update interval, the peak request rate increases exponentially making very short ACRL update intervals expensive. For our PKI model, if ACRL update interval is 1 minute, the peak request rate is around 5000/sec, a high value. To manage high requests rates, we use delta ACRLs. A delta ACRL contains a digitally signed revocation list containing information about new revocations that occurred since the issuance of the ACRL that the delta ACRL refers to (see Fig.3). The role of a delta ACRL is very similar to that of a traditional delta CRL. Every time a delta ACRL is issued, the CRL issuer also issues a complete CRL. The signature bytes of the complete CRL is included in the delta ACRL using its signature bytes extension. Using the delta ACRL and earlier issued ACRLs, a client

Table 2. ACRL Performance. As the ACRL update interval $T \rightarrow 0$, the combined peak bandwidth $\rightarrow$ 12.5 KB/sec and the total data transferred $\rightarrow$ 1.03 GB/day.

ACRL issuance interval (min)	Size of ACRL (bytes)	Size of ACRLRI (bytes)	Peak request rate (/sec)	Combined peak bandwidth (KB/sec)	Total data transferred (GB/day)
60	410.0	100.0	101.9	14.4	0.97
30	360.0	50.0	184.6	13.4	1.00
15	335.0	25.0	351.0	13.0	1.02
10	326.7	16.67	517.5	12.8	1.02
5	318.3	8.33	1017.4	12.7	1.03
1	311.7	1.67	5017.1	12.6	1.03

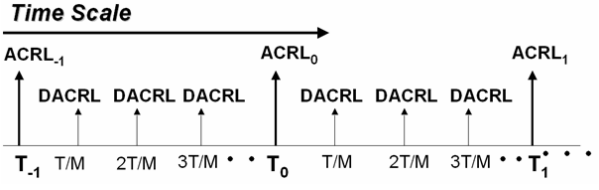


Fig. 3. Delta ACRL issuance timeline

can construct the complete CRL issued at the same time as the delta ACRL. Unlike ACRLs, delta ACRLs do not carry `CRLNumber` or a list of revoked expired certificates. As with ACRLs, a client need not download a delta ACRL in its entirety. It just needs the revocation list present in the latest delta ACRL (delta ACRL `RevocationInfo`), the signature bytes of the complete CRL (issued along with the latest delta ACRL) and the ACRL `RevocationInfo` of all intermediate ACRLs. Using these attributes, the client can construct the latest complete CRL.

Let $M-1$ delta ACRLs be issued in one ACRL issuance interval (T), one every T/M interval. Let T_0 be current time, $ACRL_0$ be the most recent ACRL issued at time T_0. RDA_j is the request rate for j^{th} delta ACRL and RA_{-i} the request rate for the ACRL `RevocationInfo` ($ACRLRI_{-i}$) issued i ACRL intervals earlier. The size of the j^{th} delta ACRL (S_{DACRL}) is given by (18). The request rate for the j^{th} delta ACRLRI is given by (19). Clients that have not obtained the most recent ACRL `RevocationInfo` ($ACRLRI_0$) and any earlier ACRL `RevocationInfo` ($ACRLRI_{-i}$) request them, the request rate for $ACRLRI_{-i}$ issued i ACRL intervals earlier given by (20). Assuming that the first ACRL was issued q intervals earlier and at steady-state q is a very large number, the combined request rate and combined peak request rate for all ACRL (including delta ACRL) data structures are given by (21) and (22) respectively. The total bandwidth (T_{BW}) is a summation of the bandwidth contributions from all downloaded ACRL data structures (23). For our PKI model, if ACRLs are issued every ten minutes, the peak bandwidth (P_{BW}) occurs when the first delta ACRL is issued (24). The total data downloaded in one ACRL update interval

is given by (25). As we reduce the ACRL update interval to shorter values, the peak bandwidth and total ACRL data downloaded (in one day) are still given by (16) and (17).

The delta ACRL performance scheme for different delta ACRL issuance intervals is shown in Table 3 (as applied to our PKI model with ACRLs issued every 10 minutes). Using delta ACRLs, we can issue ACRLs at intervals which does not overload the ACRL distribution server with high request rates, e.g. ACRLs can be issued every 10 minutes (the peak request rate for such issuance being around 520/sec, which can be easily managed by off-the-shelf servers). So delta ACRLs can be issued very frequently e.g. 1 minute, 30 seconds providing efficient online certificate status without imposing high network costs.

$$S_{DACRL}(j,T) = S_{AH} + S_E(jT/M)r_{revoke}\ ,\ j = 1\ to\ M-1\ . \tag{18}$$

$$RDA_j(t,T) = Nve^{-vt}\ (0 \le t < T/M)\ . \tag{19}$$

$$RA_{-i}(j,t,T) = Nve^{-v(t+jT/M+iT)}\ (0 \le t < T/M)\ . \tag{20}$$

$$\begin{aligned}\text{Combined Request Rate, } RA(t,T) &= RDA_j(t,T) + \lim_{q\to\infty}\sum_{i=0}^{q-1} RA_{-i}(j,t,T)\\ &= Nve^{-vt} + Nve^{-v(t+jT/M)}/(1-e^{-vT}).\end{aligned} \tag{21}$$

$$\text{Peak Combined Request Rate, } P_{RA}(T) = Nv + Nve^{-v(T/M)}/(1-e^{-vT})\ . \tag{22}$$

$$\begin{aligned}T_{BW}(j,t,T) &= (S_{SB} + S_E(jT/M)r_{revoke})RDA_j(t)\\ &\quad + \lim_{q\to\infty}\sum_{i=0}^{q-1}(\text{Size of } ACRLRI_{-i})RA_{-i}(j,t,T)\\ &= (S_{SB} + S_E(jT/M)r_{revoke})Nve^{-vt}\\ &\quad + Nve^{-v(t+jT/M)}S_{AE}r_{revoke}T/(1-e^{-vT})\ .\end{aligned} \tag{23}$$

$$P_{BW}(T) = Nv\left(S_{SB} + S_E(T/M)r_{revoke} + S_{AE}r_{revoke}Te^{-vT/M}/(1-e^{-vT})\right)\ . \tag{24}$$

$$\begin{aligned}&ACRLData_T(T)\\ &= \lim_{q\to\infty}\sum_{j=1}^{M-1}\sum_{i=0}^{q-1}\left(\begin{array}{c}(\text{Size of } DCRLRI_j + S_{SB}) \times \text{No. of Reqs. for } DCRLRI_j\\ + \text{Size of } ACRLRI_{-i} \times \text{No. of Reqs. for } ACRLRI_{-i,j}\end{array}\right)\\ &= NS_{AE}\ r_{revoke}T + N(1-e^{-vT/M})\,(MS_{SB} + 0.5S_E r_{revoke}T(M-1))\ .\end{aligned} \tag{25}$$

Compared to traditional CRL schemes, ACRLs have two minor disadvantages. Firstly, the ACRL distribution server needs to store all issued ACRLs. In our PKI model with ACRLs issued every 10 minutes, and each ACRL approximately 325 bytes, 16.3 Mbytes of ACRLs need to be stored every year. Since computer

Table 3. Delta ACRL performance for ACRLs issued every 10 minutes. As the ACRL update interval $T \rightarrow 0$, the combined peak bandwidth $\rightarrow$ 12.8KB/sec and the total data transferred $\rightarrow$ 1.05 GB/day.

Delta ACRL issuance interval (seconds)	Combined peak request rate (/sec)	Combined peak bandwidth (KB/sec)	Total data transferred (GB/day)
300	534.6	12.8	1.04
120	545.1	12.8	1.05
60	548.7	12.8	1.05
30	550.5	12.8	1.05

memory is extremely cheap, this cost is negligible. Secondly, the ACRL peak request rate is considerably higher than any other CRL scheme especially at short ACRL update intervals. But as shown earlier in this section, ACRL request rates can be reduced to manageable levels using delta ACRLs.

4 ACRL vs. OCSP

Currently the most popular online certificate status protocol is OCSP [2]. A client issues a status request to an OCSP responder and suspends acceptance of the certificate until the responder provides a definitive response. The primary objective of OCSP is to provide online certificate status at relatively (compared to traditional CRL schemes) low network loads. In this section, we show that our ACRL scheme has many advantages over OCSP. To compare, we assume that fresh certificate validation information is required every minute. So delta ACRLs are issued every minute and ACRLs every 10 minutes resulting in 12.8 KB/sec peak bandwidth and 1.05 GB/day of ACRLs downloaded. Assuming that certificate validation requests arrive independent of each other, the sustained OCSP request rate is Nv (for our PKI model, this is 34.7 requests/sec). The size of a typical OCSP response is 250 bytes (RSA 1024bit) resulting in sustained bandwidth of 8.5 KB/sec. The total OCSP response data downloaded per day is 0.7 GB/day. These values are just marginally better than our ACRL scheme. With ACRL caching, ACRLs might even perform better than OCSP. So the general opinion that only OCSP has low bandwidth costs is no longer valid.

OCSP has high operational costs. Firstly, since every OCSP response is digitally signed, the online OCSP responder is a trusted server. Operating trusted high availability online servers is expensive. Moreover, replicating OCSP responders means replicating signing keys: this increases the vulnerability of signing keys. Secondly, as each OCSP response is digitally signed, OCSP responders are particularly vulnerable to denial of service (DOS) attacks. Thirdly, the OCSP signer should either be the CA, a trusted responder or a CA designated responder [2]. If a signing key other than the CA's signing key is used, additional mechanisms are needed to determine the OCSP signing key's validity status. Lastly, OCSP has high computational demands since every OCSP response is digitally

signed. On the other hand, ACRLs can be created on offline (and more secure) computers and uploaded onto cheap online un-trusted servers making ACRL distribution inexpensive and less vulnerable to DOS attacks. Since ACRLs are not created on a per-request basis, the number of digital signatures created is much lesser compared to OCSP, making ACRLs computationally light-weight.

Complete CRLs constructed from ACRLs are identical to complete CRLs issued by the CRL issuer. The integrity of locally constructed complete CRLs is protected by the CRL issuer's signature, so CRLs can be cached, replicated and shared with any other entity. Moreover, unlike OCSP which provides certificate status on a per-request basis, complete CRLs (constructed using ACRLs) contain a complete list of all revoked un-expired certificates. Since ACRLs are delta CRLs with extensions, our scheme can be easily integrated into existing CRL based schemes unlike OCSP.

5 Conclusion

We have presented a simple yet highly effective ACRL based certificate status scheme applicable even to large PKIs. Clients have no need to download complete CRLs, clients download much smaller ACRLs and construct complete CRLs locally resulting in significantly reduced network costs. Unlike earlier CRL schemes, clients never download revocation information they already possess - ACRLs contain only revocation updates. Using our PKI model, we show that ACRLs significantly out-perform all existing CRL schemes. Contrary to existing opinion, we have presented an online certificate status scheme using ACRLs which has significant advantages over OCSP [2]. A major practical advantage of our ACRL scheme is its backward compatibility: ACRLs are delta CRLs with X.509 extensions and hence can easily be integrated into existing X.509 based certificate schemes.

References

1. ITU-T. "Information technology - Open systems interconnection - The directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509 (V4), 2000.
2. M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.
3. D. A. Cooper, "A model of certificate revocation", Proceedings of the 15th annual computer security applications conference, December 1999. http://csrc.nist.gov/pki/PKImodels/.
4. D. A. Cooper, "A more efficient use of delta CRLs", Proceedings of the IEEE symposium on security and privacy, May 2000. http://csrc.nist.gov/pki/PKImodels/.

Online/Offline Signatures and Multisignatures for AODV and DSR Routing Security

Shidi Xu, Yi Mu, and Willy Susilo

School of Information Technology and Computer Science
University of Wollongong, Australia
{sdx86, wsusilo, ymu}@uow.edu.au

Abstract. Efficient authentication is one of important security requirements in mobile ad hoc network (MANET) routing systems. However, the digital signature enabled approach is too costly for MANET due to the computation overheads. In addition, due to the diversity of different routing protocols, a universal scheme that suits all MANET routing systems does not exist in the literature. Specifically, an authentication scheme for the AODV routing is believed to be not suitable to the DSR routing. In this paper, we first introduce an efficient ID-based online/offline scheme for authentication in AODV and then provide a formal transformation to convert the scheme to an ID-based online/offline multisignature scheme. Our scheme is *unique*, in the sense that a single ID-based online/offline signature scheme can be applied to both AODV and DSR routing protocols.

Keywords: MANET, Routing, Online/offline, signature, Multisignature.

1 Introduction

The security technology deployed in the existing MANET is very weak [9]. Several well-known MANET routing protocols such as DSR [6] and AODV [10] were designed without a security consideration. Consequently, MANET routing systems face a number of security threats, from basic spoofing attacks to more complex rushing attacks. Providing full-scale security to MANET with a low computational overhead and bandwidth consumption becomes an open problem.

The security deployment to MANET is stunted by cryptographic techniques. The nature of the network requires low computational overheads, whereas existing authentication methods, such as digital signatures, are too expensive to apply. In [14], an ID-based online/offline signature scheme was proposed to provide a solution to this problem. However, this scheme is not universally applicable to all the MANET routing protocols in the sense of achieving the best efficiency. We found that the online/offline signature scheme for AODV routing protocol *does not* help for securing DSR, because of their difference in packet processing operations. To date, constructing an authentication scheme that is applicable for both AODV and DSR is remaining an interesting open problem.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 99–110, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Our Contribution. We are motivated to provide a universal authentication scheme for MANET routing protocols. We firstly introduce an identity-based (or ID-based, for short) online/offline signature scheme suitable for AODV protocol and then transform this scheme to an ID-based multisignature scheme which is suitable for the DSR protocol. We give the generic construction and the concrete constructions, and analyse the security and efficiency of the resulting scheme. Since the online/offline signature based authentication scheme for AODV protocol has been discussed in [14], in this paper, we only concentrate on providing an applicable authentication scheme for DSR protocol using above transformation.

Organisation of the paper. The rest of the paper is organised as follow. In section 2, we define the notion of the ID-based online/offline signature schemes and the ID-based multisignature scheme. Then we give the generic construction from the ID-based online/offline signature scheme to the ID-based multisignature scheme. In section 3, we present our concrete implementation of these signature schemes. We also prove the security and analyse the efficiency of the schemes. In section 4, we introduce the basics of DSR protocol and describe the application. Finally, we conclude the paper.

2 Generic Constructions

In this section, we review the definition of the ID-based online/offline signature scheme and accountable subgroup multisignature scheme and their security requirements. We also provide the generic construction of ID-based multisignature scheme based on the ID-based online/offline signature scheme.

2.1 ID-Based Online/Offline Signature Scheme

Definition 1. *ID-based online/offline digital signature scheme $\mathcal{DS}$ is comprised of five polynomial time algorithms:* IO_ParamGen, IO_Ext, IO_OffSign, IO_OnSign, *and* IO_Verify.

IO_ParamGen. *The master key and parameter generation algorithm, is a probabilistic algorithm that on input a security parameter 1^k, outputs a master key $IOSK^*$ and a parameter list params.*

IO_Ext. *The signing key issuing algorithm, is a deterministic algorithm that on input a user's identity id and a master key $IOSK^*$, returns a pair of matching public and secret keys $(iopk_{id}, iosk_{id})$.*

IO_OffSign. *The offline signing algorithm, is a probabilistic algorithm that on input a parameter list params and a signing key $iosk_{id}$, outputs an offline signature S.*

IO_OnSign. *The online signing algorithm, is a probabilistic algorithm that on input a message m and an offline signature S, returns an online signature σ.*

IO_Verify. *The verification algorithm, is a deterministic algorithm that on input a message m, a user's identity id, a parameter list params, an offline signature S, and an online signature σ, returns 1 (*accept*) or 0 (*reject*).*

The security of the online/offline signature can be defined as followed.

Definition 2. *An identity based online/offline signature is said to be existentially unforgeable under chosen-message attacks if no probabilistic polynomial time adversary has a non-negligible advantage in this game:*

1. *The challenger $\mathcal{A}$ runs the setup algorithm to generate the system parameters and sends them to the adversary $\mathcal{F}$.*
2. *The adversary $\mathcal{F}$ performs the following queries:*
   - ***Key Extraction Query*** $\mathbf{O}^{\mathbf{IOS}}_{\mathbf{Ext}}$*: $\mathcal{F}$ produces an identity ID and receives corresponding secret key D_{ID}.*
   - ***Offline Signing Query*** $\mathbf{O}^{\mathbf{IOS}}_{\mathbf{OffSign}}$*: $\mathcal{F}$ produces an identity ID, and receives an offline signature generated by offline signing oracle using the secret key corresponding to ID.*
   - ***Online Signing Query*** $\mathbf{O}^{\mathbf{IOS}}_{\mathbf{OnSign}}$*: $\mathcal{F}$ produces a message m, and receives a online signature generated by online signing oracle. The online signature is corresponding to the offline signature.*
3. *After a polynomial number of queries, $\mathcal{F}$ produces a tuple $(ID^*, m^*, S^*, \sigma^*)$ of identity ID^*, whose secret key was never asked in key extraction query. Besides, the pair (ID^*, m^*) was never asked in online/offline signing queries.*

The success probability of winning the above game is defined by $\mathsf{Succ}^{EF-IOS-CMA}_{\mathcal{A}}(\ell)$. An online/offline signature scheme is secure if the success probability of above attack is negligible.

$$\mathsf{Succ}^{EF-IOS-CMA}_{\mathcal{A}}(\ell) \leq \epsilon,$$

where ϵ is negligible.

2.2 ID-Based Multisignature Scheme

Multisignature schemes, since firstly introduced by Itakura and Nakamura [5], have been extensively studied in the literature. However the first formal definition of multisignature scheme was provided by Micali et al. [8]. Their scheme, named Accountable Subgroup Multisignatures (ASM), enables any subgroup G_{Sub} of a given group G of potential signers, to sign a message efficiently, so that the signature provably reveals the identity of the signers in G_{Sub} to any verifier.

According to Micali et al, we extend the definition of ASM to ID-based ASM.

Definition 3. *An ID-based multisignature consists of four components. We assume that the group G_{Sub} consists of L signers.*

AM_ParamGen. *The master key and parameter generation algorithm, is a probabilistic algorithm that on input a security parameter 1^k, outputs a master key $AMSK^*$ and a parameter list params.*

AM_KeyGen. *The signing key issuing algorithm, is a probabilistic algorithm that on input a subgroup G_{Sub}, a user's identity id and a master key $AMSK^*$, returns a pair of matching public and secret keys $(ampk_{id}, amsk_{id})$ for each user in the group.*

AM_Signing. *The signing algorithm, is a probabilistic algorithm that on input the following from each signer:*

1. *a description of subgroup* G_{Sub}
2. *the public key* $ampk_i$ *of each member in* G_{Sub}
3. *the message* m
4. *the signer's secret key* $amsk_i$

produces a signature σ *which is generated jointly by all the members of* G_{Sub}.

AM_Verifying. *The verification algorithm, is a deterministic algorithm, on input the following*

1. *a description of subgroup* G_{Sub}
2. *the public key* $ampk_i$ *of each member in* G_{Sub}
3. *the message* m
4. *the signature* σ

*outputs 1 (*accept*) or 0 (*reject*).*

The security definition of ID-based ASM can be adapted from the ID-based online/offline signature scheme.

Definition 4. *An ID-based multisignature (IBMS) of subgroup* $S \subseteq G$ *is said to be existentially unforgeable under chosen-message attacks if no probabilistic polynomial time adversary has a non-negligible advantage in producing a tuple* (σ, m, S) *such that:*

1. *The challenger* $\mathcal{A}$ *runs the setup algorithm to generate the system parameters and sends them to the adversary* $\mathcal{F}$.
2. *The adversary* $\mathcal{F}$ *performs the following queries:*
   - ***Key Generation Query*** $\mathbf{O}^{\mathbf{AM}}_{\mathbf{KGN}}$*:* $\mathcal{F}$ *produces an identity* ID *of the uncorrupted player in* S *and receives corresponding secret key* D_{ID} *and its temporary signing commitment* S *for current signing session.*
   - ***Signing Query*** $\mathbf{O}^{\mathbf{AM}}_{\mathbf{Sign}}$*:* $\mathcal{F}$ *produces a message* m*, and receives a signature generated by signing oracle using the secret key corresponding to* ID.
3. *After a polynomial number of queries,* $\mathcal{F}$ *produces a tuple* (m^*, σ^*, S^*) *such that*
   - σ^* *is a valid signature on the message* m *by the subgroup* S *of players.*
   - *there exists an uncorrupted player* $P^* \in S$ *who has never been asked by* $\mathcal{F}$ *to execute the signing query on* m *and* S.

The success probability of winning the above game is defined by $\mathsf{Succ}^{EF-IMS-CMA}_{\mathcal{A}}(\ell)$. *An ID based multisignature scheme is secure if the success probability of the above attack is negligible. In other words,*

$$\mathsf{Succ}^{EF-IBMS-CMA}_{\mathcal{A}}(\ell) \leq \epsilon,$$

where ϵ *is negligible.*

2.3 Generic Construction of IBSM from IOS

We observe the similarity between online/offline signature and multisignature: the offline signing phase does not involve any message in computation, therefore the resulting offline signature together with the signer's identity can be used as the public key for verifying online signature, which in turn can be treated as the signature in multisignature scheme. We provide the generic construction of multisignature scheme based on identity based online/offline signature scheme (Figure 1).

AM_ParamGen (1_k)
  $(IOSK^*,\ params) \leftarrow$ IO_ParamGen(1^k)
  $AMSK^* \leftarrow IOSK^*$
  return $(AMSK^*,\ params)$
AM_KeyGen $(G_{Sub},\ id,\ AMP_{pub})$
  $(iosk_{id},\ iopk_{id}) \leftarrow$ IO_Ext(id, AMSK*, params)
  $amsk_{id} \leftarrow iosk_{id}$
  $ampk_{id} \leftarrow iopk_{id}$ return $(ampk_{id},\ amsk_{id})$
AM_Signing $(m,\ G_{Sub},\ amsk_{id})$
  $C_{id} \leftarrow$ IO_OffSign(id, $\mathsf{iosk_{id}}$, params)
  $\sigma_{id} \leftarrow$ IO_OnSign(m, id, S, params, $\mathsf{amsk_{id}}$)
  $\widetilde{\sigma} \leftarrow \Sigma^{G_{Sub}}(\sigma_{id})$
  $\widetilde{C} \leftarrow \Sigma^{G_{Sub}}(C_{id})$
  return $(\widetilde{\sigma},\ \widetilde{C})$
AM_Verifying $(m,\ G_{Sub},\ \widetilde{\sigma},\ \widetilde{C})$
  $b \leftarrow$ IO_Verify(m, $\mathsf{G_{Sub}}$, params, $\widetilde{\sigma}$, $\widetilde{\mathsf{C}}$)
  return b

Fig. 1. Generic Construction from IOS to IBMS

Theorem 1. *The ID-based multisignature scheme is secure only if the corresponding ID-based online/offline signature scheme is existentially unforgeable against chosen-message attacks.*

Note: Due to the lack of space, we omit the security proof, but we refer the reader to the full version of this paper [13].

3 The Concrete Schemes

In this section, we provide a concrete ID-based online/offline signature scheme, and transform this scheme into the multisignature scheme using the above general construction. We also prove the security of these two signature schemes.

4 Cryptographic Tools: Bilinear Pairings

Let $\mathbb{G}_1$ be a cyclic additive group generated by P, with a prime order q, and $\mathbb{G}_2$ be a cyclic multiplicative group with the same prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a map with with the following properties:

1. Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1, a, b \in \mathbb{Z}_q^*$;
2. Non-degeneracy: There exists $P, Q \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$;
3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$;

We call such bilinear map as an admissible bilinear pairing. The problem considered in the additive group $\mathbb{G}_1$ is:

- Computational Diffie-Hellman Problem (CDHP): For $a, b \in \mathbb{Z}_q^*$, given P, aP, bP compute abP.

Above system parameters can be obtain through running the **GDH Parameter Generator** [4] $\mathcal{IG}$ which takes a security parameter $k \in \mathbb{Z}^+$ as input, runs in polynomial time in k, and outputs a prime number q, the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ of order q, and the description of an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

Definition 5. *The advantage of an algorithm $\mathcal{A}$ in solving CDHP in group $\mathbb{G}$ is*

$$\mathsf{Adv}_{\mathcal{A}}^{CDH} = \Pr[\mathcal{A}(P, aP, bP) = abP : a, b \stackrel{R}{\leftarrow} \mathbb{Z}_q^*]$$

where the probability is over the choice of a and b, and the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}(t, \epsilon)$-breaks CDHP in $\mathbb{G}$ if $\mathcal{A}$ runs in time at most t, and $\mathsf{Adv}_{\mathcal{A}}^{CDH} > \epsilon$.

4.1 Online/Offline Signature Scheme

Our scheme involves four algorithms: System Setup, ID Extract, Offline Signing, Online Signing and Verify.

Setup. Given $\mathbb{G}_1$ and its generator P, pick a random $s \in \mathbb{Z}_q^*$, and set $P_{pub} = sP$. Choose a cryptographic hash function $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The system parameters are (P, P_{pub}, H_0, H_1). The master key is s. H_0 and H_1 behave as random oracles.

Extract. Given an identity ID, the algorithm computes $D_{ID} = sH_0(ID)$ and outputs it as the private key related to ID corresponding to $Q_{ID} = H_0(ID)$.

OffSign. Given a secret key D_{ID}, pick random $r, x \in \mathbb{Z}_q^*$, output the offline signature pair (S, R), where $S = D_{ID} - xP_{pub}$, $R = rP$.

OnSign. Given a message m and offline signature S, compute the online signature as $\sigma = H_1(m)r + x$. The resulting signature is a triple (S, σ, R).

Verify. Given a signature tuple (S, σ, R) of a message m for an identity ID, check whether $(P_{pub}, S + \sigma P_{pub}, P, Q_{ID} + H_1(m)R)$ is a valid Diffie-Hellman tuple.

Signing algorithms satisfy the requirement of online/offline signature as the actual message signing takes only one hash. The size of our signature is $2 \log_2 \rho + \log_2 q$, in which ρ stands for the safe length of GDH group $\mathbb{G}_1$.

4.2 Security Analysis

To prove our scheme is existentially unforgeable under adaptive chosen-message attack, we use Libert and Quisquater's proof technique [7].

Theorem 2. *In the random oracle model, if a probabilistic polynomial time forger $\mathcal{F}$ has an advantage ε in forging an online/offline signature with running time t and asking H_0,H_1,key extraction oracle and online/offline signing oracle q_{H_0}, q_{H_1}, q_e and q_s times respectively, then the CDH problem can be solved with an advantage*

$$\varepsilon' > (\frac{1}{q_e} \cdot (1 - \frac{1}{q_e + 1})^{q_e+1})(\varepsilon - \frac{q_S(q_{H_1} + q_S) + 1}{2^k})$$

with running time $t' < t + (q_{H_0} + q_e + 2q_s)t_m$, where t_m is the time to compute a scalar multiplication in $\mathbb{G}_1$.

Note: Due to the lack of space, we omit the security proof, but we refer the reader to the full version of this paper [13].

4.3 ID-Based Multisignature Scheme

We adopt our ID-based online/offline signature scheme to the ID-based multisignature scheme according to our generic construction. The resulting scheme consists of four algorithms: system setup, key generation, signing and verifying.

Setup. Given $\mathbb{G}_1$ and its generator P, pick a random $s \in \mathbb{Z}_q^*$, and set $P_{pub} = sP$. Choose a cryptographic hash function $H_0 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$. The system parameters are (P, P_{pub}, H_0, H_1). The master key is s. H_0 and H_1 behave as random oracles

KeyGen. For each player P_i $(1 \leq i \leq L)$ in G, given an identity ID_i, the algorithm computes $D_{ID_i} = sH_0(ID_i)$ and outputs it as the private key related to ID_i corresponding to $Q_i = H_0(ID_i)$.

Signing. Each player P_i $(1 \leq i \leq L)$ in G pre-computes the following:

1. randomly choose $x_i, r_i \in \mathbb{Z}_q^*$
2. compute the signing commitment for the current session as $C_i = D_i - x_i P_{pub}$, $R_i = r_i P$, and $U_i = x_i P$
3. broadcast (C_i, R_i, U_i) to all the players.

Suppose the players in a subgroup $S = P_{i_1}, ..., P_{i_l}$ wish to jointly sign a message m. Upon receiving (C_i, R_i) from all the other players, each of them does the following:

1. verify the received public key by checking the equality of the equation:

$$e(C_i, P) = e(Q_i - U_i P, P_{pub})$$

2. if the equality holds, compute $\widetilde{C} = \sum_{j=1}^{l} C_j = \sum_{i=1}^{l} D_j - \sum_{j=1}^{l} x_j P_{pub}$
3. compute $\widetilde{R} = \sum_{j=1}^{l} R_j = \sum_{j=1}^{l} r_j P$

4. compute the signature as
   (a) each signer computes the signature $\sigma_j = H_1(m)r_j + x_j$ and broadcasts to all the signer P_{i_j} $(1 \leq j \leq l)$
   (b) upon receiving all the σ_j, each signer computes $\widetilde{\sigma} = \sum_{j=1}^{l} \sigma_j = H_1(m) \sum_{j=1}^{l} r_j + \sum_{j=1}^{l} x_j$.

   The resulting multisignature for message m is $(\widetilde{\sigma},\ \widetilde{C},\ \widetilde{R})$. To further reduce the signature size, we combine $\widetilde{\sigma}$ and $\widetilde{C}$ to obtain a new parameter $\widetilde{V}$ by

$$\widetilde{V} = \widetilde{C} + \widetilde{\sigma} P_{pub}$$

   The final signature is a pair $(\widetilde{V},\ \widetilde{R})$.

Verifying. The multisignature can be verified by all the group members who possess the pair $(\widetilde{C}, \widetilde{R})$. Given signature $\widetilde{\sigma}$, commitment $(\widetilde{pk}, \widetilde{R})$ and message m, check wether $(P_{pub}, \widetilde{V}, P, \sum_{j=1}^{l} Q_j + H_1(m, \widetilde{R})\widetilde{R})$ is a valid Diffie-Hellman tuple.

4.4 Security Analysis

We still start from assuming the existence of a forger $\mathcal{F}$ and an attacker $\mathcal{A}$, and initialise the system public key as $P_{pub} = aP$. Since the target subgroup we are supposed to attack contains one uncorrupted signer ID_* (we obtain the secret keys of all the other corrupted signers), $\mathcal{A}$ only needs to simulate $P_{uncorrupted}$ during key generation and signing. A big difference to the previous proof is that instead of letting $\mathcal{A}$ flip a coin to decide the corresponding identity is to be attack or not, we calculate the probability of getting a fixed identity by using Cha-Cheon's ID attack [1]. This probability can be used to replace δ.

Theorem 3. *In the random oracle model, if a probabilistic polynomial time forger $\mathcal{F}$ has an advantage ε in forging an ASM with running time t and asking H_0,H_1,key extraction oracle and signing oracle q_{H_0}, q_{H_1}, q_e and q_s times respectively, then the CDH problem can be solve with an advantage*

$$\varepsilon' > ((1 - \frac{1}{q})\frac{1}{q_{H_0}})(\varepsilon - \frac{q_S(q_{H_1} + q_S) + 1}{2^k})$$

with running time $t' < t + (q_{H_0} + 4q_e)t_m$, where t_m is the time to compute a scalar multiplication in $\mathbb{G}_1$.

Note: Due to the lack of space, we omit the security proof, but we refer the reader to the full version of this paper [13].

4.5 Efficiency Comparison

To compare the efficiency, we assume the safe length of GDH group $\mathbb{G}_1$ is ρ and the order of multiplicative group is q. We analyze the efficiency of signature schemes in relation to four indicators: signature size, pre-computation cost, signing cost, verification cost and problem based. We define the pre-computation phase to include all the operations taken irrelevant to the message to be signed.

The signing phase only contains the operation aiming at the message. The signing cost and pre-computation cost are justified in terms of the elliptic curve scalar multiplications (ESM), or exponentiations being used. The verification cost is justified by counting the number of pairings being used. We also assume the multisignature is generated by n signers.

We choose five existing ID-based multisignature schemes, in which three of them use bilinear pairings and the other two are based on RSA. Besides our scheme (**IBMS**, based on IOS), another four schemes include: **SOK-IBMS** [11] proposed by Sakai et al., **CZK-IBMS** [3], the ID-based blind multisignature scheme proposed by Chen et al, based on Cha-Cheon scheme [1], **WH-IBMS** [12] proposed by Wu and Hsu, **CLL-IBMS** [2] proposed by Chang et al.

We firstly look at the comparison between single ID-based signature schemes in **Table 1**. It is obvious that our online/offline signature scheme is efficient in online signing since no ESM needs to be performed. Two RSA based signature schemes are efficient in signing and verification, but signature sizes are apparently larger than others. The comparison of the ID-based multisignature schemes

Table 1. ID-based Signature Efficiency Comparison

	Signature Size	Pre-comp.	Signing Cost	Verification Cost	Problem
SOK-IBS [11]	$2\log_2 \rho$	1 ESM	1 ESM	2 pairings	CDHP
Cha-Cheon [1]	$2\log_2 \rho$	1 ESM	1 ESM	2 pairings	CDHP
IOS	$2\log_2 \rho + \log_2 q$	2 ESM	0 ESM	2 pairings	CDHP
WH-IBS [12]	$\log_2 N$	N/A	1 expon.	2 expon.	RSA
CLL-IBS [2]	$\log_2 N$	N/A	1 expon.	3 expon.	RSA

is listed in **Table 2**. We can see that the Chen et al.'s scheme is very efficient in average, requiring $2n$ scalar multiplications in the pre-computation phase and the signing phase. Our scheme performs the same number of scalar multiplications ($2n$) in pre-computation phase. However, the actual signing phase needs only 1 scalar multiplication. We can draw this conclusion that our ID-based multisignature scheme preserves the advantage of its original scheme (ID-based online/offline signature scheme), which is able to shift the computational overhead to the pre-computation phase.

Table 2. ID-based Multisignature Efficiency Comparison

	Signature Size	Pre-comp.	Signing Cost	Verification Cost	Problem
SOK-IBMS [11]	$(n+1)\log_2 \rho$	n ESM	$\Sigma_{i=1}^{n} i$ ESM	3 pairings	CDHP
CZK-IBMS [3]	$2\log_2 \rho$	n ESM	n ESM	2 pairings	CDHP
IBMS	$2\log_2 \rho$	$2n$ ESM	1 ESM	2 pairings	CDHP
WH-IBMS [12]	$\log_2 q$	N/A	n expon.	$(n+1)$ expon.	RSA
CLL-IBMS [2]	$\log_2 q$	N/A	$2n$ expon.	3 expon.	RSA

5 Application to the DSR Protocol

We firstly introduce some basics of the DSR protocol and analyse its security requirements. Then we will provide the implementation of ASM over DSR.

5.1 DSR Protocol

DSR stands for dynamic source routing protocol, presented by Johnson and Maltz [6] in 1996. It is an on-demand routing protocol based on the concept of source routing, which means the initiator knows the complete hop-by-hop route to the destination. To perform DSR, each node is required to maintain a route cache which contains the topology information of the network. The route cache is consistently updated to reflect the current situation of the network.

DSR consists of two phases: route discovery and route maintenance. When a node wants to send data to another node, it firstly searches its route cache to see if there is a route to this destination. If yes, this route will be used. Otherwise, this node generates a route request packet (RREQ) which consists of a data structure called *route record* listing the IP addresses of all the intermediate nodes. This RREQ will be broadcasted to neighbours. Each of the neighbouring nodes will search its own route cache to see if there exists an active route to the destination. If not, it appends its own IP address to *route record* and rebroadcasts it to its neighbours. This process will be continued until the RREQ packet reaches the destination. The original message is not changed during the transmission (except the RREQ data length field which is a number). The resulting route will be found in the *route record*.

In replying the RREQ, the destination node generates a route reply packet (RREP) and sends it back to the initiator by two ways. It could search its route cache, use the route already existed, or perform its own route discovery. It could also simply reverse the sequence of hops in *record list*.

5.2 Installation of IBMS over DSR

We firstly define the total signers' group to include all the mobile nodes in MANET. The maximum size of the total group G should agree with the network capacity. We then define the subgroup S to include the mobile nodes involved in a routing operation. Therefore, each routing operation will accordingly form a subgroup whose maximum size equals the maximum hop count allowed by DSR protocol. Before a DSR based network is initialised, the total group is set as empty $G \leftarrow \phi$, so as to the subgroup.

To perform the ID-based authentication, we assume the existence of an offline key generation center (KGC). KGC runs the system Setup algorithm to generate all the parameters required. Each node, before entering the network, has to submit its credential to KGC. The KGC will run the key generation algorithm (KeyGen) to generate a public-secret key pair for each node. Once a node has obtained all the necessary parameters, it can start to do all the pre-computations according to signing algorithm, in order to achieve the best efficiency in signing.

When a RREQ is issued, the initiator runs the signing algorithm Signing to generate a signature over all the immutable fields. The mutable fields, the RREQ data length field and the route address field, are excluded and their values are set to 0 during the signature generation.

The RREQ along with the signature will be broadcasted to next hop neighbours. The next hop nodes will firstly run the verification algorithm Verifying to evaluate the signature validity. To run this algorithm, the verifier firstly needs to extract the IP addresses, which are also the public keys of previous hop nodes, from the RREQ. Accordingly, if a malicious node deliberately removes some IP addresses from the RREQ, the signature will not pass the verification and the route carried by the RREQ will be considered as incorrect and rejected. Therefore, by performing the signature verification, both the signature and the route are authenticated.

If the signature is valid, the verifier (the next hop neighbour) will produce a new signature over the immutable field of the original received message. This node then appends its own IP address to the RREQ and broadcasts the RREQ along with the signature. The neighbours of the third hop will perform the same operations as the neighbours of the second hop did to produce signatures over the original RREQ generated by the initiator. This process will continue until the RREQ reaches the target node. The target node, after verifying and accepting the RREQ, will respond with a RREP. This RREP will be transmitted back to the initiator along the route discovered. In this condition, the signature of the RREP will be processed the same as the RREQ.

One arguable point of using multisignature in a sequential form is that the node is able to remove itself from the path. We argue that removing itself does not make any sense. To remove itself, a node passes the routing packet to its next hop neighbour without changing anything. For example, the node M receives a route packet from node A and passes the packet to node B without adding its IP address to *route record* and increasing the value of data length field. There are two situations that could happen. Firstly, if node A is in the neighbourhood of node B and node M's behaviour actually results in a legal route which is one hop shorter. This route will be accepted by node B, or generated by node B sooner or later. On the other hand, if node A is not in the neighbourhood of node B, removing node M results in node B to receive a packet from a distant node. Since node B constantly uses acknowledge packet (ACK) to confirm the link, it will detect the illegality of this packet and finally drop it.

6 Conclusion

We introduced the notion of ID-based online/offline signature scheme and ID -based multisignature scheme. We presented a generic construction of ID-based multisignature scheme based on ID-based online/offline signature scheme. We also provided a concrete scheme of ID-based online/offline signature scheme and transformed it into the ID-based multisignature scheme using our generic construction. Our scheme is proved secure against existential forgery under adaptive

chosen message attacks based on the random oracle model assuming that CDHP problem is hard. We compared our scheme with other ID-based multisignature schemes and concluded the transformation could inherit the quick signing capability from the online/offline signature scheme. We provided the application over the DSR protocol and argue that our scheme is especially suitable for DSR where the routing messages are modified by appending IP addresses and discussed the implementation issue of the DSR protocol.

References

1. J. Cha and J. Cheon. An ID-based signature from gap-diffie-hellman groups. In *Proceedings of Public Key Cryptography - PKC 2003*, volume 2567, pages 1–24. Springer-Verlag, 2003.
2. C. Chang, I. Lin, and K. Lam. An ID-based multisignatures scheme without reblocking and predetermined signing order. In *Computer Standards and Interfaces*, pages 407–413. Elsevier Science Inc., 2004.
3. X. Chen, F. Zhang, and K. Kim. ID-based multi-proxy signature and blind multisignature from bilinear pairings. In *Proceedings of KIISC'2003*, pages 11–19, 2003.
4. D.Boneh, B. Lynn, and H. Shacham. Short signature from the weil pairing. In *Proceedings of Asiacrypt '01, Lecture Notes in Computer Sciences*, volume 2248, pages 514–532. MANET working group, 2001.
5. K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. NEC Research and Development, 1983.
6. D. B. Johnson, D. A. Maltz, and Y. C. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, 2004.
7. B. Libert and J-J. Quisquater. The exact security of an identity based signature and its applications. In *Cryptology ePrint Archive, Report 2004/102*, 2004.
8. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *Proceedings of the 8th ACM Conference on Computer and Communications Security: CCS '01*. Springer, 2001.
9. P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference: CNDS 2002*, 2002.
10. C. E. Perkins, E. M. Royer, and S. R. Das. *Ad Hoc On-Demand Distance Vector (AODV) Routing*, 2003.
11. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of Symposium on cryptography and Information Security: SCIS 2000*, 2000.
12. T. Wu and C. Hsu. ID-based multisignatures with distinguished signing authorities for sequential and broadcasting architectures. In *Applied Mathematics and Computation*, pages 349–356. Elsevier Science Inc., 2002.
13. S. Xu, Y. Mu, and W. Susilo. Online/offline signatures and multisignatures for AODV and DSR routing security full version (available upon request).
14. S. Xu, Y. Mu, and W. Susilo. An efficient authentication scheme for manet routing. In *Proceedings of the Embedded and Ubiquitous Computing: EUC 2005 Workshops*. Springer, 2004.

Towards an Invisible Honeypot Monitoring System

Nguyen Anh Quynh and Yoshiyasu Takefuji

Graduate School of Media and Governance, Keio University
5322 Endoh, Fujisawa, Kanagawa, Japan 252-8520
{quynh, takefuji}@sfc.keio.ac.jp

Abstract. Honeypot is a decoy system to trap attackers, and data capture tool is one of the components of the honeypot architecture. Being used to collect the intruder's activities inside the honeypot, this key component must be able to function as stealthily as possible, so the intruder does not know that he is under watch. Unfortunately Sebek, a de-facto tool for this purpose in the modern honeypot technology, is rather easy to detect, even with unprivileged right access. This paper proposes to use Xen Virtual Machine to deploy honeypot, and takes the advantage introduced by Xen to fix some of the outstanding problems of Sebek. We present a design and implementation of a Xen-based system named *Xebek* as a solution. While Xebek provides similar features as Sebek does, our system is more "invisible" and harder to defeat. The experimental results also demonstrate that Xebek is more flexible, while the reliability and efficiency are significantly improved over its counterpart.

1 Introduction

Honeypot ([1]) is a computer system with the purpose: to lure attacker in order to gather information about threats. These collected information is used to better understand threats, how they are evolving and changing, in order to counter those threats in the best way possible. If applying the honeypot technology properly, we can discover the novel attack patterns and unknown security holes. Honeypot also helps to study the attacker's motives.

The modern honeypot offers high level of interaction for the intruders, and consists of 3 key components:

- Data control: this component is used to contain the intruder's activities and ensure that he does not cause any harm to other production systems outside the honeypot.
- Data capture: a honeypot must capture all the activities within the honeypot, including the information entered and left the system.
- Data collection: the gathered information got from the capture component must be securely and secretly forwarded to a central data server for analysis and archiving.

Sebek ([2]) is a de-facto, widely-used tool in current honeypot technology. Sebek architecture consists of 2 key components: a kernel module run on the honeypot system, and a central server to collect data. The first component, Sebek kernel module, serves as the data capture tool, and can capture intruder's activities in the honeypot. It also serves as a part of the data collection component: the collected data is then transferred

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 111–122, 2006.
© Springer-Verlag Berlin Heidelberg 2006

by this module to the server component (*sebekd*) running on a central machine, and then analyzing process is taken there with some utilities provided with Sebek package.

One of the vital requirements of the data capture component is that it must function as stealthily as possible, so the intruder never knows that he is being observed. Unfortunately researchers have pointed out many methods to detect Sebek's presence, and some of them do not even require privileged access.

This paper analyses some of the outstanding problems of Sebek, then proposes a novel approach based on Xen Virtual Machine ([3]) to address these problems. We will present a new architecture together with an implementation of a data capture tool named Xebek. While Xebek still has the capture ability of Sebek, it is much more "invisible", far more reliable and the efficiency also significantly increases.

The rest of this paper consists of 5 sections: The 2nd section outlines some problems with current capture tools (specifically focus on Sebek). The 3rd section proposes Xebek architecture together with its implementation in detail. The next section evaluates our solution with some benchmarks, while the section 5th discusses some methods to harden Xebek. Section 6th summaries related works, and finally conclusion closes this paper in the 7th section.

2 Sebek Problems

Technically, Sebek kerne module stays inside the honeypot kernel, and this module tries to capture all the data that can disclose the intruder's activities. The collected data include keystrokes, input and output data from file system or network socket. Sebek even patches the kernel to hide its presence from system users, so the intruder would have hard time to discover its existence.

Unfortunately while many efforts are put on to make Sebek stealthy, the attacker can still easily uncover and defeat it. Here are 8 outstanding problems of Sebek:

(1) Unprivileged local user can run a simple command that generates big input or output data, such as the *dd-attack* method proposed in [4], then checks to see if the network performance is decreased with ping command. The reason is that the data captured from all the input and output are secretly forwarded to the central *sebekd* server via network stack, and this activity may lead to congest network if the data is enormous.
(2) Though Sebek tries to patch the network stack to hide the actual network statistics when using network layer to send data, we can still compare the internal statistics got from inside the kernel with what the system shows to user-space ([4]). The difference will disclose the presence of Sebek.
(3) Sebek inserts a kernel module into the system, and this module could be listed in the kernel module list (with *lsmod* command on Linux). Though we can try to hide it (like using the method proposed by adore-ng [5]), hidden kernel module can still be found with a brute-force scanning technique [6].
(4) Sebek replaces some system-calls with its own functions. Paper [4] exploits this fact to propose another way to discover Sebek: we just need to check if the address of these system-calls are at abnormal places in the memory. If that is the case, chances are Sebek is present in the kernel.

(5) Sebek replaces some system-calls with the code that packs the data into UDP packets and send them out to the *sebekd* machine. This change significantly increases the time to complete the system-calls, and the difference (more than 184 times in the case of open system-call as in our experimental) can be easily recognized from user-space by unprivileged user.
(6) After detecting Sebek, the intruder can remove it by recovering the original system-call (for example with *unsebek.c* tool in [7]). The fact that Sebek is a kernel module makes it easier to do that.
(7) Sebek sends the captured data to the central server via network. If the intruder has a sniffer (such as tcpdump [8]) installed at the right place in the network, he will see these data and easily figure out that the system he has penetrated is a honeypot.
(8) The central server must expose to the network to receive data sent from the honeypot. That will tempt the intruder to attack this server to bring down this fundamental component of our honeypot. This is not a theory, but the actual threat: Paper [9] proposes such a method, in which *sebekd* will be taken over if it uses a libpcap library with buffer overflow bug.

As we see, there are too many problems with the current Sebek, and they all make honeypot less attractive solution for security practices.

3 Xebek Solution

Traditionally, honeypots have been physical systems on a dedicated network, with multiple physical machines to monitor and collect logging data from the honeypots. The requirement resources posed by honeypot prevent it become a common network security solution. Fortunately, the advent of virtual machines such as Xen has made setting up honeypots far easier. Instead of a set of physical machines, the honeypots is now the Xen virtual machines running on the same physical machine: each virtual machine can play a specific role: data control, data capture or data collection.

Our solution Xebek is based on Xen, and takes some advantages provided by Xen to address the outstanding problems which Sebek currently experiences.

3.1 Xen Virtual Machine

Xen is an open source virtual machine that allows to partition a machine to support the concurrent execution of multiple operating systems (OS). Commodity OS (now officially Linux, FreeBSD, NetBSD are supported) can run on Xen with small changes to the kernel. Technically, Xen is a thin layer of software above the bare hardware, and Xen exposes a virtual machine abstraction that is slightly different from the underlying hardware. In Linux, Xen introduces a new architecture called *xen*, which is very similar to x86 architecture. The virtual machine (VM) executing on Xen are modified (at kernel level) to work with xen architecture. Running on top of Xen, VM is called Xen domain, or domain in short. A privileged special domain named Domain0 (or Dom0 in short) always runs. Dom0 manages other domains (called User Domain, or DomU in short), including jobs like start, shutdown, reboot, save, restore and migrate them between physical machines.

3.2 Xebek's Goals and Approaches

Xebek is designed with the aim to overcome 8 problems experienced by Sebek we discussed above.

1. The first goal of Xebek is to capture data as Sebek does on honeypot system. In our Xen diagram the honeypot system runs on a DomU, and all the activities happened inside this domain must be captured : this includes keystrokes, input and output from file system and socket. To do that, Xebek employs the same techniques as Sebek does by modifying kernel system-calls. But while Sebek works as a module, we propose Xebek as kernel patch, so we do not need to worry about hiding kernel module as Sebek does, and it is also more difficult for the intruder to remove Xebek from kernel. With this trick, we are not worried any more about hiding kernel module as Sebek does, and it is also more difficult for the intruder to remove Xebek from kernel. Therefore, the problem (3) of Sebek is addressed with our approach, while the problem (6) is much more relieved (we will discuss further on this later). Moreover, since we no longer patch the system-call, our solution fixes the problem (4) of Sebek.
2. Another mission for Xebek is to eliminate the problem of leaving many traces while sending data to through the network stack with Sebek. To solve this trouble, Xebek is designed so all the data is forwarded to the central server via shared memory, and it never uses network stack like Sebek does. Specifically, we will instead take the advantage of Xen to send the data out via shared memory. Since all the domains run on the same physical machine, they can share memory with each other. Thanks to Xen intercommunication mechanisms, we can establish a shared memory between DomU, the virtual machine we are trying to run Xebek on, and Dom0. DomU puts all the gathered data in the shared memory, then notifies Dom0 to pick up them. Obviously with this scheme, data is no longer sent out through the network stack, thus the process becomes more quietly, stealthily, and subsequently the intruder cannot detect Xebek by looking at network traffic like he can with Sebek. This trick solves the problems (1), (2), (5) and (7) Sebek currently suffers.

   In addition, this approach brings one more merit: data is sent via shared memory (but not network stack and the unreliable UDP protocol employed by Sebek), the overall reliability and efficiency is significantly increased.
3. Xebek should harden central logging server: With the strategy of exchanging data between DomU and Dom0 via shared memory, we run a daemon process in Dom0 to pick up logging data forwarded out by DomU. Because all the communication is done via shared memory and other Xen communication mechanisms, the whole process is not carried out on the network. Consequently the daemon process is not necessarily exposed on the network like *sebekd* does, hence it is not vulnerable to the direct attack from outside. Thus our approach is able to address the problem (8) of Sebek.

   Because Xen provides strong isolation between DomU and Dom0, even if the intruder knows that he is under observation, he cannot access or modify the logging data kept in Dom0. This advantage still stands even if he somehow gains the ultimate privileges of root user.

4. Xebek must be flexible, so the administrator can disable or enable it as he desires at run-time.

All of those goals and approaches lead us to the architecture for Xebek as followings.

3.3 Xebek Architecture

Xebek consists of 3 main components: The Xebek device in DomU, which plays as a data capture tool (*xebekU*); the logging recorder in Dom0, which plays as a data collection daemon (*xebekd*); and utilities in Dom0 (including keystroke extractors, database up-loader and others). The overall architecture of Xebek is outlined as in Fig.1.

xebekU: *xebekU* is a kernel code Xebek put in kernel-space of DomU. This code patches the system-calls (such as open, close, read, write, socket,...) to gather the data coming in and out of the system. The collected data is then delivered to Dom0 via a shared memory between DomU and Dom0.

To be flexible, *xebekU* can be disabled and enabled by an instruction sent from Dom0's user-space. When inactive, it costs no overhead in the DomU.

xebekd: *xebekd* is a logging recorder running in user-space of Dom0 to record data sent from *xebekU*. This daemon process patiently waits for the notifications on the new data from *xebekU*. If it detects that the new data arrived, it gets the data from the shared memory between Dom0 and DomU above, then saves the data into separate logging files for each domain respectively.

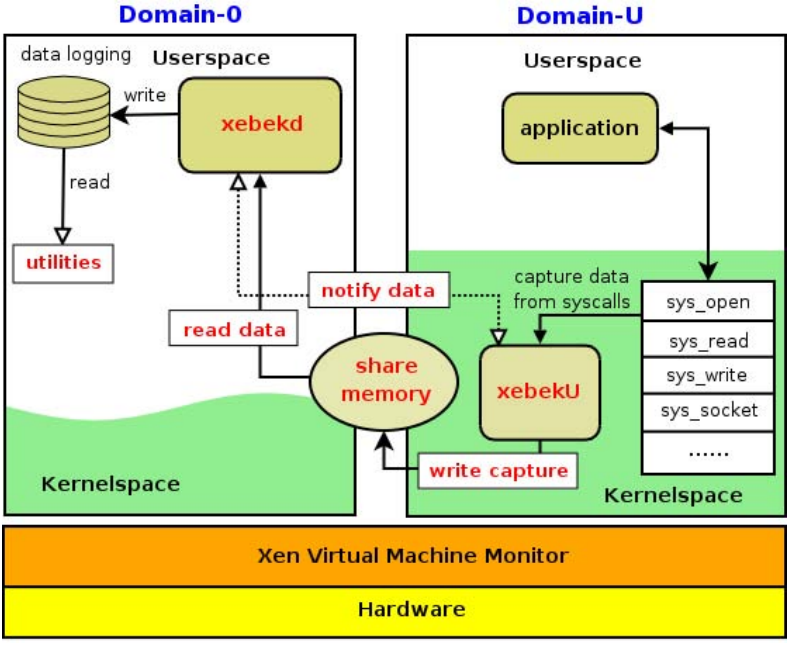


Fig. 1. Xebek architecture

Add-on utilities: Xebek has some utilities to extract interested data from the logging files of *xebekd*. We intend to provide what Sebek provides with Sebek package, so it is easier for people familiar with Sebek to adopt Xebek. For the time being, a tool to extract keystrokes from logging data and another tool to upload data to a SQL server are available.

3.4 Xebek Implementation

At the moment Xebek is only implemented in Linux, because other Os-es (like FreeBSD and NetBSD) are not ready for Xen 3.0, the most advanced Xen version we are working on, yet. So in this part we will present Xebek's implementation specifically for Linux environment. The same techniques can be applied for others, however.

xebekU: *xebekU* is the kernel code run in DomU. One of the most important jobs of *xebekU* is to gather the data from I/O system-calls such as read, write, socket,.... To do that these system-calls in DomU's kernel are modified, so the patched system-calls deliver their data to *xebekU*. With each of these system-calls, we define corresponding type, and the type is recorded with the logging data, so we can distinguish these data when analyzing them later. Some of the types are: OPEN, READ, WRITE, SOCKET,... (for *sys_open*, *sys_read*, *sys_write*, *sys_socket*...). Those records are saved in a structure of *xebek_packet* type (see Fig.2), in which we store also information such as the owner's uid, process ID, file descriptor and inode number of the corresponding file. The actual data follows the packet. This format is compatible with Sebek logging format, and that is one of our important targets.

Since DomU and Dom0 run on the same physical machine, they can share memory with each other. When *xebekU* initializes, it allocates some memory for sharing (the amount of shared memory is configurable at runtime - by default is 1 page, which is equivalent to 4KB on x86 systems). This shared memory is then used to store the collection data fetched from the kernel mentioned above.

To communicate with *xebekd*, *xebekU* assigns an *event-channel* port to send notifications to *xebekd*. After that, *xebekU* informs *xebekd* the value of the physical address of

```
struct xebek_packet {
 uint16_t magic;
 uint16_t event;
 uint16_t version;
 unsigned int fd;
 uid_t uid;
 pid_t pid;
 pid_t ppid;
 unsigned long inode;
 uint32_t size;
 char comm[12];
 struct timeval time;
} __attribute__((packed));
```

Fig. 2. The *xebek_packet* structure

the shared memory got in the above step, together with the event-channel port. At this moment, the event-channel is not established yet, so *xebekU* writes these information to *xenstore* via *xenbus* interface ([10]). The xenstore nodes where *xebekU* saves these data are *xebek/ring-mfn* and *xebek/event-channel* nodes in the xenstore home of the domain.

Shared Memory Structure and xebekd's Internal Buffer. The shared buffer must be read and written at the same time by *xebekU* and *xebekd*. These conflicted activities can causes the unpleasant race issues. This directs us to a decision: the shared buffer should be designed as a ring buffer. Ring buffer is special data structure with 2 heads: one for reading and one for writing, and these heads can wrap-around when they reach the end of the buffer. Writing data to buffer will take away some spaces, but reading from the buffer will release some spaces, and the free space then might be used for another written request later.

The internal buffer of *xebekd* also uses the same data structure, so at the same time it can be written to with data from the shared memory, and read out by the recording thread.

Logging Recorder. *xebekd* is a multiple thread daemon runs in user-space of Dom0 to gather collection data forwarded from *xebekU*. Those data are put in the shared memory between Dom0 and DomU. In order to do that, *xebekd* must do the following jobs:

(i) **Manage the DomUs**
To mange DomUs, *xebekd* must detect when a DomU notifies *xebekd* that it wants to exchange information with *xebekd*, so *xebekd* must be aware of domain initialized event. *xebekd* registers *xenstore watch* ([10]) for the event *@introduceDomain*. Whenever a domain appears, this watch notifies *xebekd*, and *xebekd* setups a new xenbus watch to detect the written event of *ring-mfn* and *event-channel* nodes in the xenstore home of that domain. The watch allows *xebekd* to detect when the domain writes down *ring-mfn* and *event-channel* to the xenstore. Once *xebekd* gets such a notify, it allocates some internal structures to manage the domain, maps the memory shared by DomU (with given *ring-mfn* is the physical address of shared memory), and bounds to the *event-channel* received above. From then on, *xebekd* is able to handle and exchange notifications with DomU via event-channel.

(ii) **Pick up data sent from *xebekU* and save them to logging files**
To pick up data delivered from *xebekU* and save them to logging files, we use 2 separate theads: a main thread is used to pick up data, and a *worker* thread is used to save the data to file-system. The main thread quietly waits for the notification about the new data from *xebekU*. When it detects that the new data arrives, it reads the data out from the corresponding shared memory, then copies them to a *host-buffer* allocated by *xebekd* when initializing (this *host-buffer* is of the ring buffer format declared above). While the shared memory size should be limited (because that is the memory allocated by DomU's kernel, and kernel memory is a precious limited resource), this memory can be much bigger (we set aside 32KB for this area). After collecting data from the shared memory, the main thread wakes the *worker* up to do its job. Fig.3 below outlines the diagram of the whole process.

Regarding the *worker* thread, this thread simply waits to be waken up by the main thread (via *pthread_cond_signal()* function), and reads the data from *host-*

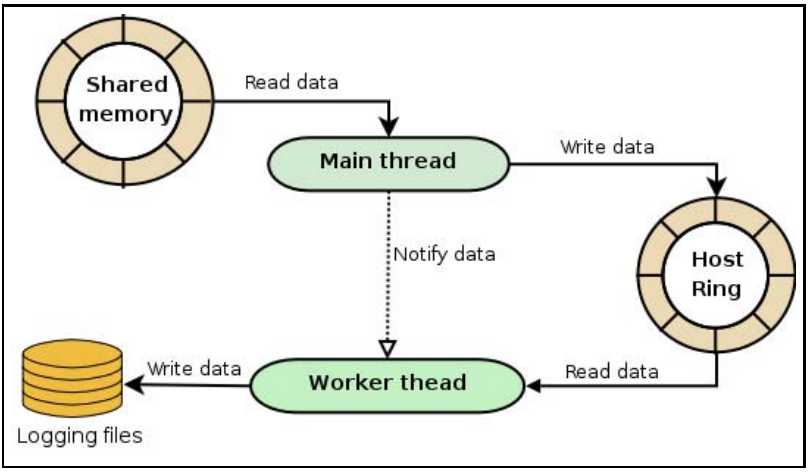


Fig. 3. The *xebekd* work-flow

buffer. The *xebek_packet* structure is extracted out, and the record is saved to the logging files on file-system, separately for each domains.

(iii) **Other stuffs**
Finally, to be more flexible, *xebekd* has one option (*-a*) to enable or disable *xebekU* of a certain domain, so the administrator can start or stop *xebekU* logging from any domain any time he wants.

Add-On Utilities. Similarly to Sebek, Xebek provides some tools to extract data out from the logging files output from *xebekd*. For the time being we provides 2 kits: *xebek_key* (written in C language) is the tool to extract keystrokes from the logging data, and *xebek_upload* (written in Python language) to upload the data to a MySQL database server for analyzing later. We adapted *Walleye* ([11]), the Web-based interface analysis tool of Sebek, for Xebek to investigate the collected data. Fig.4 describes the scheme of the above utilities.

All in all, our Xebek system is around 3600 lines of C and Python source code. Because we propose Xebek as a patch to the DomU's kernel, we try to isolate the change to the kernel, so Xebek is as less intrusive as possible: totally our code alters the kernel code only 267 lines, including comments. Table 1 shows the detailed modification (in number of lines) to the Xen-tinized Linux kernel 2.6.12.5.

Table 1. Number of lines of modification to Xen-tinized Linux kernel 2.6.12.5

File name	Number of altered lines
kernel/fork.c	54
fs/open.c	21
fs/read_write.c	148
net/socket.c	44

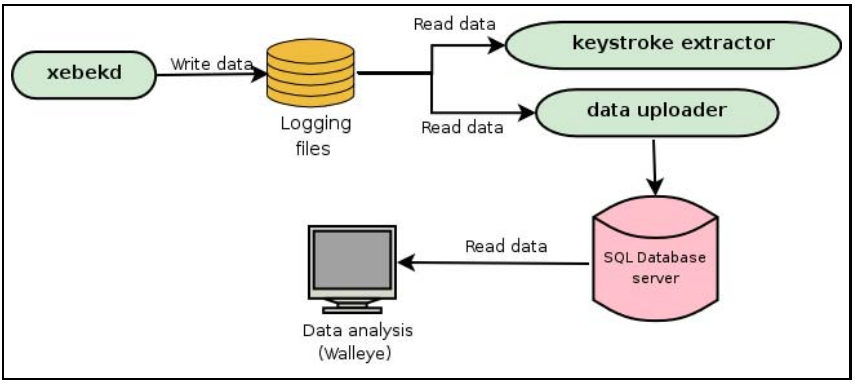


Fig. 4. Xebek utilities

4 Evaluation

To evaluate the efficiency of Xebek over Sebek, we run several benchmarks on the native OS (DomU without any capture tool), DomU's kernel loaded with Sebek, and DomU's kernel with Xebek patch, then compare the latency measurements. We focus on the latency because it is an effortless method to defeat honeypot (proposed by [12]). Especially many micro-benchmarks can be run without the privileged access right, and the intruder can employ them to discover our capture tool.

In the evaluation, we employed LMbench to run the latency tests on *read*, *write*, *open*, *fork* and *socket* system-calls, and call them READ, WRITE, OPEN, FORK, TCP and UDP test respectively: READ test measures how long it takes to read one byte from */dev/zero*, WRITE test measures the time to write one byte to */dev/null*, while OPEN measures the time to open and then close a file, and FORK test measures the time necessarily to create a new process and then shutdown it. Finally TCP and UDP measures interprocess communication latencies of TCP and UDP protocols. These tests cover all the modifications Sebek/Xebek does to the kernel.

We run each test 10 times, and then gets the average result. The configurations of the domains in the benchmarks are as below:

Dom0: Memory: 384MB RAM, CPU: Pentium3 600MHZ, IDE HDD: 40GB, NIC: 100Mbps
DomU: Memory: 128MB RAM, file-backed swap partition: 512MB, file-backed root partition: 2GB

All the domains in the tests run Linux Ubuntu distribution (version Breezy Badger), with the latest updates.

The Sebek software we use in the evaluation is 3.1.2b, which is the latest Sebek release as of this writing. This version of Sebek supports Linux kernel 2.6.x, and is compiled and loaded into the memory as a kernel module. In the test we configured Sebek so it modifies the sock system-call to track down INET family protocol (this configuration is to measure the impact on the TCP and UDP benchmarks). Besides,

though Sebek also patches the *write* system-call, we turned off this feature because there are some stable problems as recommended by the authors of Sebek.

The TCP/UDP test needs one machine functioning as a server. We established a private LAN of 100Mbps bandwidth, which consists of only 2 physical machines: one is the Xen machine, the other is a machine that runs the TCP and UDP servers of LMbench (command *lat_tcp* and *lat_udp* with *-s* option). The TCP/UDP benchmarks measures the latency to send a block data of 64MB between DomU and the server machine. During the evaluation we shutdowns all the unnecessary services to ensure that the result is not affected by outsiders.

Table 2 shows the result of the benchmarks - all the numbers are in microseconds.

Table 2. Latency benchmark of native kernel, kernel with Sebek and kernel with Xebek

	Native kernel	**Sebek kernel**	**Xebek kernel**
OPEN	8.19417	1509.07384	9.72046
READ	1.22187	972.64953	1.96883
WRITE	1.10639	1.11343	1.82222
FORK	900.38095	900.43333	900.42144
TCP	842.25625	1276.56287	1004.91216
UDP	1050.99115	1100.26256	1085.24115

We can see that for Sebek kernel, except the WRITE test is not different from the native kernel (the reason is that we chosen not to active the Sebek *write* system-call, as we mentioned above), all the other benchmarks demonstrate that Sebek incurs too much overhead: 184.16 times of the native kernel in the case of OPEN benchmark, and 796.07 times of the native kernel in the case of READ benchmark. The overheads of the TCP test and UDP test are 51.56% and 4.68%, respectively. The reason of the bad performance of Sebek on Xen in the first 3 cases seems to be the inefficient way Sebek allocates and manages kernel memory while it is functioning. We are investigating the solution to improve Sebek on this aspect.

Meanwhile Xebek kernel costs significantly less overhead: 18.62% in OPEN test, 61.13% in READ test and 64.69% in WRITE test, while TCP and UDP latency overheads are around 19.31% and 3.25%. In all the cases the latencies made by Xebek are lower than Sebek.

To best of our knowledge, there is no published benchmark of Sebek, and we consider the above evaluation a minor contribution of this paper.

5 Discussion

While Xebek is able to observe DomUs, we do not intend to watch the control domain (Dom0), because Dom0 is the trusted domain. The administrator must protect the Dom0 at all cost, as if the intruder takes over Dom0, the game is over: he can do anything he likes to other DomU. Normally it is a good idea to run Dom0 without network address, so the outsider have less chance to attack it.

To prevent the intruder from disabling Xebek, all the path to the kernel memory should be prohibited, as the intruder might somehow get the root access in DomU and

use that privilege to access the kernel internal and modify it to disable Xebek. In order to prevent this problem, DomU's kernel should be compiled with */dev/{kmem,mem,port}* removed ([13]), and the ability of loading kernel module at run-time should be eliminated, too. This can lead to some objections: the honeypot becomes too restrictive, and the attacker might suspect. But we argue that this kind of harden environment is increasingly popular, and it should be expected by the attacker on any industrial systems.

While we tried hard to make Xebek as covert as possible, unfortunately there is still a weak point: the attacker can rely on the latency created by Xebek to detect its presence. Further optimize our code to mitigate this problem is our on-going research topic.

Some might argue that the intruder can easily detect Xebek if he recognizes that he is in a Xen machine. But that is not really we are worried about: Xen community is working to merge Xen into Linux kernel, and once the job is completed, Xen is available everywhere. Thus a machine running Xen is not necessarily a honeypot, but can be a production system as well.

6 Related Works

Honeypot is the hot topic on security research field. Many papers focus on applying honeypot to improve defense system or to trap malwares. The honeypot can be broken down into 2 kinds: low-interaction and high-interaction type.

The low-interaction honeypots have limited interaction: they normally work by emulating services and operating systems. Attacker activity is limited to the level of emulation by the honeypot.

The honeynet ([14]), the high-interaction honeypot, which is the main research topic of this paper. A honeynet may contain one of more honeypots, and Sebek plays a key-role in a honeypot, with the job is to capture the intruder's activities. Though Sebek is a popular tool in the honeypot community, there are few papers that discuss the weak points of honeypot or propose methods to improve Sebek, which are related to the topic of this paper.

In [4], [12], [7] and [9] J.Corey, M.Dornseif and T.Holz have pointed out some problems with honeypot, especially with Sebek, and several methods were proposed to defeat it (more details are in the section 2 of this paper). Our paper investigates all the current outstanding problems of Sebek, and proposes Xebek as the solution to address or mitigate them.

7 Conclusions

This paper analyses the "visible" problem of Sebek, a data capture component of the modern honeynet architecture. We then proposed the design and implementation of a solution based on Xen virtual machine to eliminate some problems of Sebek. We demonstrated that Xebek can be used instead of Sebek in Xen environment, and if being installed in a strict manner, Xebek is stealthier, harder to detect, even with privileged user. Besides, we also believe that Xebek is more flexible, effective and reliable than Sebek.

For the time being, Xebek only works for Linux-based domains. We plan to provide support for other Os-es such as FreeBSD, NetBSD once these ports are working stably on Xen.

References

1. Balas, E., Viecco, C.: Towards a Third Generation Data Capture Architecture for Honeynets. In: The 6th IEEE Information Assurance Workshop. (2005)
2. The Honeynet Project: Know your enemy: Sebek. http://www.honeynet.org/papers/sebek.pdf (2003)
3. Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Pratt, I., Warfield, A., Barham, P., Neugebauer, R.: Xen and the art of virtualization. In: Proceedings of the ACM Symposium on Operating Systems Principles. (2003)
4. Dornseif, M., Holz, T., Klein, C.: NoSEBrEaK - Attacking honeynets. In: The 5th Annual IEEE Information Assurance Workshop. (2004)
5. stealth: adore-ng rootkit. http://stealth.7530.org/rootkits/ (2004)
6. madsys: Advanced incident response tool. http://sourceforge.net/projects/airt-linux/ (2005)
7. Corey, J.: Local honeypot identification. http://www.phrack.org/unofficial/p62/p62-0x07.txt (2003)
8. TCPdump project: tcpdump/libpcap tool. http://www.tcpdump.org (2005)
9. Corey, J.: Advanced honeypot identification and exploitation. http://www.phrack.org/ unofficial/p63/p63-0x09.txt (2004)
10. Xen project: Xen interface manual. http://www.cl.cam.ac.uk/Research/SRG/netos/xen/readmes/interface/interface.html (2005)
11. The Honeynet Project: Know Your Enemy: Honeywall CDROM Roo. http://www.honeynet.org/papers/cdrom/roo/ (2005)
12. Holz, T.: Detecting honeypots and other suspicious environments. In: Proceedings of the 6th IEEE Information Assurance Workshop. (2005)
13. sd: Linux on-the-fly kernel patching. http://www.phrack.org/show.php?p=58&a=7 (2002)
14. The Honeynet Project: Know your enemy: Honeynets. http://www.honeynet.org/papers/honeynet/ (2005)

Adaptively Secure Traitor Tracing Against Key Exposure and Its Application to Anywhere TV Service

Kazuto Ogawa[1], Goichiro Hanaoka[2], and Hideki Imai[2]

[1] Science & Technical Research Laboratories,
Japan Broadcasting Corporation
1-10-11 Kinuta, Setagaya-ku, Tokyo 157-8510, Japan
ogawa.k-cm@nhk.or.jp
[2] Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology
1102 Akihabara Daibiru, 1-18-13 Sotokanda, Chiyoda-ku, Tokyo 101-0021, Japan
{hanaoka-goichiro, h-imai}@aist.go.jp

Abstract. We propose a secure *traitor tracing scheme against adaptive key exposure* (TTaKE) which contains the properties of both a traitor tracing scheme and a forward secure public key cryptosystem. It is constructed by using a polynomial with two variables to generate the user secret keys. This scheme enables identification of at least one of traitors. It employs temporary secret keys by updating them. Moreover, we show the way how the building blocks of the TTaKE can be applied to anywhere TV service. Its structure fits current broadcasting systems. The system can improve content broadcasting/distribution services and it enables users to obtain the service at the places where there are not their own STBs/receivers.

1 Introduction

Background: Several methods of protecting copyrighted work from illegal distribution have been developed. Content providers, prior to content distribution, distribute decoders (STB) which contain secret keys for content decryption. A content provider sends encrypted content to each user, who then decodes it with his STB. In such a system, if a malicious user (traitor) extracts the secret keys from his STB, they can make a pirate decoder. Several traitor tracing methods have been developed to protect content against such violations [2, 3, 5, 8, 9, 10, 11]. When a pirate decoder is found, these methods are used to check the secret keys in the decoder and trace the traitor.

On the other hand, users require obtaining services anywhere they want, which is called anywhere TV service. Though they generally set their STBs and get the services at their houses, and it is hard to extract decryption keys from STBs. Consequently, they cannot get the services outside. It is possible, if users can take along their decryption key with them, and providers' risk due to key exposure is taken into account. Various countermeasures have been developed to minimize

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 123–135, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the damage resulting from key exposure [4, 6, 7]. They employ key updating and give secret keys a temporal property.

To make anywhere TV service secure, properties of both traitor tracing and robustness against key exposures are needed. The traitor tracing property deters traitors to make a pirate decoder and the robustness against users' accidental key exposure is necessary for minimizing the damage to content providers.

TTaKE: In this paper, we propose *a secure traitor tracing scheme against adaptive key exposure attacks* (TTaKE). We first define a TTaKE scheme, and then construct a TTaKE scheme which is semantically secure against chosen plaintext attacks under the assumption of the Decision Diffie-Hellman problem. Its traceability is based on the difficulties of solving the discrete logarithm problem. This scheme combines the properties of a traitor tracing scheme and a forward secure public key cryptosystem. It enables identifying users from their secret keys and tracing at least one of the traitors. Each user's secret key is updated using two kinds of keys, and the user therefore can update his secret key, only when he has the two keys.

In the initial phase, a polynomial with two variables is produced. Then one public key and multiple initial secret keys as well as master keys are made. In the TTaKE scheme, there are multiple secret keys for one public key. Each user's master key is stored in his physically secure device (PSD) such as a smart card and his initial secret key is stored in his portable memory device (PMD) such as a USB memory device.

The users cannot access the data stored in PSDs directly and hence master keys are resistant to compromise. On the other hand, when the user decrypts contents, his secret key is picked up from his PMD and is put into an insecure device such as PC. Then the secret key is likely to be exposed. To minimize damage of such key exposure, we employ user secret key updating. The user contacts with the master key stored in the PSD, when he updates his secret key. We consider the case in which the PSD is not fully trusted, and the master key is used to generate a partial key. After generating a partial key, the user calculates a new secret key using both previous secret key and a partial key. When he does not have both a previous secret key and a master key, it is impossible to generate a new secret key.

We assume that the secret key is stored in a PMD and is carried out. Thus the secret key is copied easily, and it is possible to make *a pirate decoder (PD)*. The TTaKE scheme that we propose here can identify illegal users, who make PDs.

TTaKE Application to Anywhere TV Service: We show how the building blocks can be used for applying the scheme to anywhere TV Service. The service is an improved service of a conventional broadcasting service and is the one that users can obtain the service at any location where they want. Therefore users would have to take along their decryption keys with them.

In the conventional broadcasting system, a STB has a PSD or tamper resistant module (TRM). When we apply TTaKE to anywhere TV service, we

consider its compatibility with conventional broadcasting system. We thus use this PSD/TRM for storing a master key and generating a partial key. Moreover, considering users' convenience, users can carry their secret keys stored in a PMD. Furthermore, owing to the TTaKE, the system enables to trace illegal users and to minimize damage from key exposure.

2 Definitions

2.1 Model

TTaKE is a public key system in which there is a unique encryption key and multiple decryption (secret) keys.

One public key is registered, which will not be changed for as long as the service continues. Different master keys $SK_1^*, \cdots, SK_N^*$ and initial secret keys $SK_{1,0}, \cdots, SK_{N,0}$ are distributed to N users. These master keys SK_u^* are stored in each user's PSD. The user secret key $SK_{u,t}$ is updated and stored outside the PSD. The user can receive the service in any location by using $SK_{u,t}$ stored in a PMD which a user can carry with him.

First, the period during which the service will continue is set, and then this period is divided into T number of small periods. A user will have to periodically update the secret key.

The distributed content is encrypted and constructed as a header $Head$. The time index t is added to $Head$ and the distribution data is formatted as $<t, Head>$.

If authorized users collude and make an illegal decoder and the number of colluders is at most k, at least one of them should be traceable. Furthermore, even if m secret keys of the T periods have been exposed, there is no exposure of the other keys' information.

The user inputs t into the PSD, when a secret key is updated. The PSD outputs a partial key $SK'_{u,t}$. The user calculates $SK_{u,t}$ using $SK_{u,t-1}$ and $SK'_{u,t}$. If it is hard to extract SK_u^* from the PSD and to obtain any information about SK_u^* from $SK_{u,t}$, the user secret key is secure.

We formally describe this model as follows.

Definition 1. A TTaKE scheme consists of the following six polynomial time algorithms (Gen,Upd*,Upd,Enc,Dec,TT).

Gen: The public key and user secret key generation algorithm. This is a probabilistic algorithm which takes as input a security parameter, s, the total number of users, N, the maximum number of colluding users (traitors) per one service period, k, which include true colluders and honest users who expose all of their secret keys, the total number of time periods, T, and the maximum number of time periods, m, at which honest users' secret keys are exposed unintentionally. It returns a public key, PK, users' master key, $SK_1^*, \cdots, SK_N^*$, users' initial key, $SK_{1,0}, \cdots, SK_{N,0}$, and secret information to trace users, f.

Upd*: The device key updating algorithm. This is a deterministic algorithm which takes as input the time period index, t $(1 \leq t \leq T)$, and SK_u^*. It returns a user partial key, $SK'_{u,t}$.

Upd: The user key updating algorithm. This is a deterministic algorithm which takes as input t, $SK'_{u,t}$, and a user previous secret key, $SK_{u,t-1}$. It returns a user current secret key, $SK_{u,t}$.
Enc: The encryption algorithm. This is a probabilistic algorithm which takes as input PK, t, and a message, M. It returns a ciphertext, $C :=< t, Head >$.
Dec: The decryption algorithm. This is a deterministic algorithm which takes as input $SK_{u,t}$, and C. It returns M, or a special symbol, $\perp$. We require the following for all messages, $\mathsf{Dec}(SK_{u,t}, (\mathsf{Enc}(t, PK, M)) = M$.
TT: The user tracing algorithm. This is a deterministic algorithm which takes as input PK, f, and $\{SK_{p_i,t}\}_{i=1,\cdots,k_l} \subseteq \{SK_{u,t}\}_{u=1,\cdots,N}$ where $1 \leq k_l \leq k$. It returns one of the suspected traitors' IDs, $p \in \{p_i\}_{i=1,\cdots,k_l}$.

We should consider a traitor tracing against a linear attack proposed in [11]. We describe it briefly in section 4.1. Moreover, we should consider a black box traitor tracing, but we will study it in future.

Next, we define the pirate decoders.

FPD: The pirate decoder with full functionality. This must correctly decrypt all the valid ciphertext generated by Enc for all service periods.
TPD_t: The pirate decoder with limited functionality. This must correctly decrypt all the valid ciphertext generated by Enc for a service period t.

2.2 Security

We address security definition of a TTaKE scheme. The TTaKE scheme is considered secure if

- for a given FPD or TPD_t (PD), TT of the TTaKE can detect one of the IDs of authorized users who collude to make a PD, or who carelessly exposed their temporal keys that were then used to make a PD.
- without a PD, an adversary cannot obtain any information on the distributed content for the target time period t.

First we define (k, N, m, T)-traceability which addresses security against forgery of a PD such that any of colluders cannot be traced.

Each user has his identical secret key. Since, when traitors make a PD with their own secret keys, it is easy to identify one of traitors, traitors want to make a PD that holds the secret key that is not theirs. Hence, when we consider the traceability, we consider that traitors generate a different secret key from their own ones. Intuitively, we say a TTaKE scheme is (k, N, m, T)-*traceable* if k colluders cannot forge any $SK_{u,t}$ as long as none of honest users' keys for a time period t is exposed and the number of time periods that an honest user's key is exposed is at most m.

For a given public key PK, an adversary adaptively chooses k colluders and m time periods such that at least one honest user's key is exposed. Then, the adversary decides a target time period t^*, and a victim u^*. To formally model key exposure attacks, we give an adversary access to two types of key exposure oracle $Exp_u(\cdot)$ and $Exp_t(\cdot)$. $Exp_u(\cdot)$ takes as input a user ID u and returns all

of the time periods' user secret keys, $\{SK_{u,t_j}\}_{j=1,\cdots,T}$. $Exp_t(\cdot)$ takes as input a time period index t and returns all of users' secret keys of the time period t, $\{SK_{u_i,t}\}_{i=1,\cdots,N}$. We allow that the key exposure requests of the adversary may be made adaptively and in any order.

Definition 2. Let Π=(Gen,Upd*,Upd,Enc,Dec,TT) be a TTaKE scheme. Let A be an adversary. Define the success probability of guessing the value of SK_{u^*,t^*} as follows:

$$
\begin{aligned}
&Succ_{A,\Pi}(s,k,N,m,T) \stackrel{\text{def}}{=} Pr[\\
&\quad (PK, SK_1^*, \cdots, SK_N^*, SK_{1,0}, \cdots, SK_{N,0}, f) \leftarrow \mathsf{Gen}(1^s,k,N,m,T);\\
&\quad (u^*, t^*, SK^A_{u^*,t^*}) \leftarrow A^{Exp_u(\cdot),Exp_t(\cdot)}(PK) :\\
&\quad \forall M \in \mathcal{M}, \mathsf{Dec}(SK^A_{u^*,t^*}, \mathsf{Enc}(t^*, PK, M)) = M],
\end{aligned}
$$

where $\mathcal{M}$ is the message space. An adversary can access $Exp_u(\cdot)$ and $Exp_t(\cdot)$, and is allowed to request $Exp_u(\cdot)$ at most k times except for the target user(u^*)'s secret key and $Exp_t(\cdot)$ at most m times except for the target time period(t^*)'s secret key. Then Π is (k,N,m,T)-traceable if for any adversary $|Succ_{A,\Pi}(s,k,N,m,T)|$ is negligible.

Next, we define (m,T)-*indistinguishability* which addresses semantic security against an adversary who can adaptively obtains exposed secret keys. Similarly to the standard definition of semantic security, for a given public key PK, an adversary chooses a time period t^*, and a pair of messages with the same length M_0 and M_1, and submits them to a *left-or-right encryption oracle* which returns a challenge ciphertext $c^* := \mathsf{Enc}(t^*, PK, M_b)$ for $b \in_R \{0,1\}$. A TTaKE is considered semantically secure if any probabilistic polynomial time Turing machine can answer the correct value of b with probability at most $1/2+neg$ where neg is a negligible value. In our definition, an adversary can obtain exposed keys which he chooses, and he may use these keys for the attack with a restriction that t^* may not be identical to a valid time period of any exposed key. To formally model key exposure attacks, we give an adversary access to $Exp_t(\cdot)$. We allow that the key exposure requests of the adversary may be made adaptively and in any order. See also Def. 1 on other restrictions for the number of exposed keys with respect to m.

Definition 3. Let $\Pi = (\mathsf{Gen}, \mathsf{Upd}^*, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TT})$ be a TTaKE scheme. Let $A = (A_{find}, A_{guess})$ be an adversary. Define the success probability of guessing the value of b as follows:

$$
\begin{aligned}
&Succ_{A,\Pi}(s,k,N,m,T) \stackrel{\text{def}}{=} Pr[\\
&\quad (PK, SK_1^*, \cdots, SK_N^*, SK_{1,0}, \cdots, SK_{N,0}, f) \leftarrow \mathsf{Gen}(1^s,k,N,m,T);\\
&\quad (t^*, M_0, M_1, \sigma) \leftarrow A_{find}^{Exp_t(\cdot)}(PK);\\
&\quad b \in_R \{0,1\};\ c^* \leftarrow \mathsf{Enc}(t^*, PK, M_b);\\
&\quad b' \leftarrow A_{guess}^{Exp_t(\cdot)}(PK, \sigma, c^*) :\\
&\quad b' = b],
\end{aligned}
$$

where σ is side information obtained by A_{find}, and an adversary can access $Exp_t(\cdot)$ and is allowed to request to $Exp_t(\cdot)$ at most m times except for the target time period t^*. Then Π is (m,T)-*indistinguishable* if for any adversary $\left|Succ_{A,\Pi}(s,k,N,m,T) - \frac{1}{2}\right|$ is negligible.

Finally, we define (k,N,m,T)-*security*.

Definition 4. Let $\Pi = (\mathsf{Gen}, \mathsf{Upd}*, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TT})$ be a TTaKE scheme. Π is (k,N,m,T)-*secure* if it is (k,N,m,T)-traceable and (m,T)-indistinguishable.

When traitors make an FPD or TPD_{t^*}, it is not necessary to consider semantic security for the target time period t^*, and we consider only traitor tracing, described in Def. 2. On the other hand, it is important to consider semantic security in Def. 3. Even if an adversary gets exposed secret keys, which are valid at some periods, content of the other time periods should be safe. Totally a TTaKE scheme has (k,N,m,T)-security described in Def. 4.

3 (k,N,m,T)-Secure Traitor Tracing Scheme Against Key Exposure

We demonstrate a (k,N,m,T)-secure traitor tracing scheme against adaptive key exposure ((k,N,m,T)-TTaKE), which is based on the corrected Kurosawa-Desmedt traitor tracing scheme [8, 9] and the (m,T)-key-insulated public-key cryptosystems [7]. Below we propose a way to construct a (k,N,m,T)-TTaKE scheme.

$\mathsf{Gen}(1^s,k,N,m,T)$: Let p and q be primes such that $q \mid p-1$ where the size of $|q|$ is s and let $\mathbb{G}_q$ be a subgroup of $\mathbb{Z}_p^*$ of its order q. After this setting, all calculations are executed in $\mathbb{Z}_p$. Two generators, $g,h \in \mathbb{G}_q$, and random numbers, $x_{i,j}, y_{i,j} \in \mathbb{Z}_q$ for $i = 0,\cdots,2k-1; j = 0,\cdots,m$, are selected, and two two-variable polynomials, $f_1(u,t) := \sum_{i=0}^{2k-1}\sum_{j=0}^{m} x_{i,j}u^i t^j$ and $f_2(u,t) := \sum_{i=0}^{2k-1}\sum_{j=0}^{m} y_{i,j}u^i t^j$, are made, and a public key, $PK := (g,h,p,q,z^*_{0,0},z^*_{0,1},\cdots,z^*_{2k-1,m})$ where $z^*_{i,j} = g^{x_{i,j}}h^{y_{i,j}}$ for $i = 0,\cdots,2k-1; j = 0,\cdots,m$, is published. Then each user's master key, $SK^*_u := (x^*_{u,1},y^*_{u,1},x^*_{u,2},y^*_{u,2},\cdots,x^*_{u,m},y^*_{u,m})$, and initial secret key, $SK_{u,0} := (x^{(s)}_{u,0},y^{(s)}_{u,0})$ where $x^*_{u,j} = \sum_{i=0}^{2k-1} x_{i,j}u^i$, $y^*_{u,j} = \sum_{i=0}^{2k-1} y_{i,j}u^i$ for $j = 1,\cdots,m$, and $x^{(s)}_{u,0} = \sum_{i=0}^{2k-1} x_{i,0}u^i$, $y^{(s)}_{u,0} = \sum_{i=0}^{2k-1} y_{i,0}u^i$, are made.

$\mathsf{Upd}*(t,SK^*_u)$: A partial key, $SK^{'}_{u,t} := (x^{'}_{u,t},y^{'}_{u,t})$ where $x^{'}_{u,t} = \sum_{j=1}^{m} x^*_{u,j}(t^j - (t-1)^j)$, $y^{'}_{u,t} = \sum_{j=1}^{m} y^*_{u,j}(t^j - (t-1)^j)$, is calculated using input time index t and SK^*_u.

$\mathsf{Upd}(t,SK^{'}_{u,t},SK_{u,t-1})$: A secret key, $SK_{u,t} = (x^{(s)}_{u,t},y^{(s)}_{u,t})$ where $x^{(s)}_{u,t} = x^{'}_{u,t} + x^{(s)}_{u,t-1}$, $y^{(s)}_{u,t} = y^{'}_{u,t} + y^{(s)}_{u,t-1}$, is calculated using $SK^{'}_{u,t}$ and $SK_{u,t-1}$.

$\mathsf{Enc}(t,PK,M)$: A random number, $\alpha \in \mathbb{Z}_q$, is chosen and a header, $Head(t) := (y_g,y_h,z_{0,t},\cdots,z_{2k-1,t})$ where $y_g = g^\alpha$, $y_h = h^\alpha$, $z_{t,0} = M(\prod_{j=0}^{m}(z^*_{0,j})^{t^j})^\alpha$, $z_{i,t} = (\prod_{j=0}^{m}(z^*_{i,j})^{t^j})^\alpha$ for $i = 1,\cdots,2k-1$, is produced using PK, a message M, and t. Then a ciphertext $C :=< t, Head(t) >$ is created.

$\mathsf{Dec}(C, SK_{u,t} = (x_{u,t}^{(s)}, y_{u,t}^{(s)}))$: C is decrypted using $SK_{u,t}$. Then M is obtained through the following calculation: $M = z_{0,t} \prod_{i=1}^{2k-1} (z_{i,t})^{u^i} / y_g^{x_{u,t}^{(s)}} y_h^{y_{u,t}^{(s)}}$.

Furthermore, a traitor tracing capability is necessary to fit into the model. $\mathsf{TT}(PK, f_1(u,t), f_2(u,t), SK_{p,t})$: When an FPD or TPD_t is found, a secret key $SK_{p,t}$ is checked and one of traitors p is identified.

We describe and discuss this tracing algorithm in section 4.1.

4 Security Analysis

4.1 Tracing Traitors

It should be noticed that an FPD is considered as a set of TPD_ts for every time periods and therefore with regard to traceability it is sufficient to consider forgery of a TPD_t.

Tracing traitors: When a TPD_t is confiscated, the user identity and secret key $((u_p, f_1(u_p,t), f_2(u_p,t))$ contained in it are exposed. The exposed user u_p is a traitor.

When the traitors collude to make a TPD_t, they might try to make a TPD_t that includes a different user's identity and secret key to avoid being traced. We show that it is hard for at most k traitors to make the other users' secret keys and our proposed scheme can trace one of the k traitors, who extract their secret keys and collude to make a TPD_t. As a result, it is (k, N, m, T)-traceable as described in Def. 2.

Theorem 1. *The proposed scheme is a (k, N, m, T)-traceable assuming infeasibility of the discrete logarithm (DL) problem in $\mathbb{G}_q$.*

Proof. Here, we assume that there exists a probabilistic polynomial time adversary A that can produce a TPD_t such that none of traitors can be traced. Then, we show that it is possible to construct another adversary B which can solve the DL problem in $\mathbb{G}_q$ with non-negligible probability by using A.

For a given instance of the DL problem $(g_1, g_2 = g_1^r) \in \mathbb{G}_q^2$, B works as follows. First, B chooses $\{x_{i,j}, y_{i,j}\}_{i=0,\cdots,2k-1, j=0,\cdots,m} \in \mathbb{Z}_q$, generates two polynomials, $f_1(u,t) := \sum_{i=0}^{2k-1} \sum_{j=0}^{m} x_{i,j} u^i t^j$ and $f_2(u,t) := \sum_{i=0}^{2k-1} \sum_{j=0}^{m} y_{i,j} u^i t^j$, and calculates a public key $PK := (g_1, g_2, p, q, g_1^{x_{0,0}} g_2^{y_{0,0}}, \cdots, g_1^{x_{2k-1,m}} g_2^{y_{2k-1,m}})$. Then, B gives PK to A. When A submits a query u to $Exp_u(\cdot)$, it returns secret keys $\{SK_{u,t_j}\}_{j=1,\cdots,T}$, and when A submits a query t to $Exp_t(\cdot)$, it returns secret key $\{SK_{u_i,t}\}_{i=1,\cdots,N}$. When A outputs a secret key $SK_{u^*,t^*} := (z_1, z_2)$ for u^* and t^*, B calculates $r = -(z_1 - f_1(u^*, t^*))/(z_2 - f_2(u^*, t^*))$, and outputs r as the solution of the DL problem. □

Claim 1. *Simulation of $Exp_u(\cdot)$ and $Exp_t(\cdot)$ are perfect, and therefore, A outputs a TPD_{t^*} for u^* with non-negligible probability.*

Proof. It is clear that B knows all coefficients of $f_1(u,t)$ and $f_2(u,t)$, and hence, B can answer any user's key of any time period. □

From Claim 1, we have that A is given the same environment as a real attack.

Claim 2. *It is information theoretically impossible to obtain any information on SK_{u^*,t^*} from the answers of $Exp_u(\cdot)$ and $Exp_t(\cdot)$.*

Proof. For simplicity of the proof, we relax the restriction on $Exp_u(\cdot)$ oracle access, that is, we set the maximum number of the queries to be $2k-1$ instead of k. Without loss of generality, we assume that A asks $u_1, \cdots, u_{2k-1}$ to $Exp_u(\cdot)$ and $t_1, \cdots, t_m$ to $Exp_t(\cdot)$. Let $U := \begin{pmatrix} u_1^0 & \cdots & u_1^{2k-1} \\ \vdots & \ddots & \vdots \\ u_{2k-1}^0 & \cdots & u_{2k-1}^{2k-1} \end{pmatrix}$ and $P := \begin{pmatrix} t_1^0 & \cdots & t_m^0 \\ \vdots & \ddots & \vdots \\ t_1^m & \cdots & t_m^m \end{pmatrix}$. Also, let A and B be $(2k-1) \times (m+1)$ and $2k \times m$ matrices such that $A \begin{pmatrix} t^0 \\ \vdots \\ t^m \end{pmatrix} = \begin{pmatrix} f_1(u_1, t) \\ \vdots \\ f_1(u_{2k-1}, t) \end{pmatrix}$ and $(u^0, \cdots, u^{2k-1})B = (f_1(u, t_1), \cdots, f_1(u, t_m))$, respectively. Here, we show that there exist q different $2k \times (m+1)$ matrices D such that $UD = A$ and $DP = B$, and that these q different matrices result in q different values for $f_1(u^*, t^*)$ with a uniform distribution.

Without loss of generality, we can set $D = D_0 + D_1$ where D_0 and D_1 are $2k \times (m+1)$ matrices such that $UD_0 = A$, $D_0P = B$, $UD_1 = 0$ and $D_1P = 0$. Therefore, to prove the existence of q different D, it is sufficient to prove that there exist q different D_1 for a fixed D_0. Since $\mathrm{Rank}(U)$ is $2k-1$, there exists a non-zero $2k$ dimensional column vector $\mathbf{v}$ such that $U\mathbf{v} = 0$ and therefore, D_1 can be expressed as $(\alpha_0\mathbf{v}, \cdots, \alpha_m\mathbf{v})$ for certain $(\alpha_0, \cdots, \alpha_m) \in \mathbb{Z}_q^{m+1}$. Similarly, since $\mathrm{Rank}(P)$ is m, there exists a non-zero $(m+1)$ dimensional row vector $\mathbf{u}$ such that $\mathbf{u}T = 0$ and therefore, D_1 can be expressed as $\begin{pmatrix} \beta_0\mathbf{u} \\ \vdots \\ \beta_{2k-1}\mathbf{u} \end{pmatrix}$ for certain $(\beta_0, \cdots, \beta_{2k-1}) \in \mathbb{Z}_q^{2k}$. D_1 then can be expressed as $\gamma\mathbf{v}\mathbf{u}$ for $\gamma \in \mathbb{Z}_q$. Hence, there exist q different D according to γ. Finally, we prove the above q different D results in q different values for $f_1(u^*, t^*)$. For this, it is sufficient to prove that $\delta := (u^{*0}, \cdots, u^{*2k-1})D_1 \begin{pmatrix} t^{*0} \\ \vdots \\ t^{*m} \end{pmatrix}$ has q different values according to D_1. Since D_1 can be expressed as $\gamma\mathbf{v}\mathbf{u}$, δ can also be expressed as $\delta = \gamma(u^{*0}, \cdots, u^{*2k-1})\mathbf{v}\mathbf{u} \begin{pmatrix} t^{*0} \\ \vdots \\ t^{*m} \end{pmatrix}$. Therefore, to prove the existence of q different values for δ, it is sufficient to prove that $(u^{*0}, \cdots, u^{*2k-1})\mathbf{v} \neq 0$ and $\mathbf{u} \begin{pmatrix} t^{*0} \\ \vdots \\ t^{*m} \end{pmatrix} \neq$

0. Assuming $(u^{*0}, \cdots, u^{*2k-1})\mathbf{v} = 0$, we have $\begin{pmatrix} u^{*0} & \cdots & u^{*2k-1} \\ u_1^0 & \cdots & u_1^{2k-1} \\ \vdots & \ddots & \vdots \\ u_{2k-1}^0 & \cdots & u_{2k-1}^{2k-1} \end{pmatrix} \mathbf{v} = 0$, and thus, $\mathbf{v} = 0$. This is a contradiction. Similar to this, we can prove $\mathbf{u} \begin{pmatrix} t^{*0} \\ \vdots \\ t^{*m} \end{pmatrix} \neq 0$. Hence, we see that answers from $Exp_u(\cdot)$ and $Exp_t(\cdot)$ does not give any information on $f_1(u^*, t^*)$. Similarly to this, we can also show that any information on $f_2(u^*, t^*)$ cannot be exposed from $Exp_u(\cdot)$ and $Exp_t(\cdot)$, either. □

Claim 3. *For the given PK, there exist q possible pairs for* $(f_1(u^*, t^*), f_2(u^*, t^*))$ *even if accesses to* $Exp_u(\cdot)$ *and* $Exp_t(\cdot)$ *are allowed.*

Proof. From Claim 2, $Exp_u(\cdot)$ and $Exp_t(\cdot)$ do not give any information on SK_{u^*,t^*}. Therefore, it is sufficient to prove that A cannot find a correct $(f_1(u^*, t^*),\ f_2(u^*, t^*))$ from q possible pairs even if A has unlimited computational power. Since for all $i \in \{0, \cdots, 2k-1\}$ and $j \in \{0, \cdots, m\}$, we have $g_1^{x_{i,j}} g_2^{y_{i,j}} = g_1^{x_{i,j}+ry_{i,j}}$, and there exist q different pairs for $(x_{i,j}, y_{i,j})$ for all i and j, even if A can compute the discrete logarithm problem. In other words, A knows only $F(u,t) := f_1(u,t) + rf_2(u,t)$, but cannot specify $f_1(u,t)$ and $f_2(u,t)$ from q possible pairs of polynomials $f_1'(u,t)$ and $f_2'(u,t)$ such that $F(u,t) := f_1'(u,t) + rf_2'(u,t)$. Consequently, there also exist q different pairs of (z_1, z_2) such that $F(u^*, t^*) = z_1 + rz_2$. □

From Claim 3, we have that A outputs (z_1, z_2) uniformly from q possible pairs such that $F(u^*, t^*) = z_1 + rz_2$. Consequently, we can compute $r = -(f_1(u^*, t^*) - z_1)/(f_2(u^*, t^*) - z_2)$ unless $(z_1, z_2) = (f_1(u^*, t^*), f_2(u^*, t^*))$.

Linear Attack Traceability: Here, we briefly discuss security against forgery of TPD_t whose key does not form as (u^*, z_1, z_2). More specifically, we should consider traceability against linear attacks [11]. The linear attack for the TTaKE scheme is as follows. First k traitors make a combined secret key $SK_{c,t} := (x_{c,t}^{(s)}, y_{c,t}^{(s)})$, with their valid secret keys $\{SK_{u_{p_l},t}\}_{l=1,\cdots,k}$ for the target time period t, and extra information $U_1, \cdots, U_{2k-1}$, where $\alpha_l \in \mathbb{Z}_q$ for $l = 1, \cdots, k$, $\sum_{l=1}^k \alpha_l = 1$, $x_{c,t}^{(s)} := \sum_{l=1}^k \alpha_l x_{u_{p_l},t}^{(s)}$, $y_{c,t}^{(s)} := \sum_{l=1}^k \alpha_l y_{u_{p_l},t}^{(s)}$, $U_i := \sum_{l=1}^k \alpha_l u_{p_l}^i$ for $i = 1, \cdots, 2k-1$. With these data and C, M is decrypted as follows: $M = z_{0,t} \prod_{i=1}^{2k-1} (z_{i,t})^{U_i} / y_g^{x_{c,t}^{(s)}} y_h^{y_{c,t}^{(s)}}$.

It is proven in [12] that, when k traitors collude to make a pirate decoder with the linear attack, one of the traitors can be identified when the degree of polynomial is more than $2k-2$. In the (k, N, m, T)-TTaKE scheme, the polynomial's degree on u is $2k-1$, and it can be proven by the same proof method of [12] that TTaKE scheme can be identified one of k colluders. As a result our scheme is secure against a linear attack.

4.2 Chosen-Plaintext Security Based on DDH

Above, we showed that our proposed scheme is a (k, N, m, T)-traceable scheme. Here, we show a proof of (m, T)-indistinguishability for our proposed scheme and that overall, it is a (k, N, m, T)-secure TTaKE scheme as described in Def. 4. First we show that our proposed scheme is semantically secure against a passive adversary, assuming the difficulty of the Decision Diffie-Hellman (DDH) problem in $\mathbb{G}_q$. The assumption is that no polynomial time algorithm can distinguish with non-negligible advantage between the two distributions $D =< g_1, g_2, g_1^a, g_2^a >$ and $R =< g_1, g_2, g_1^a, g_2^b >$, where g_1 and g_2 are generators chosen at random in $\mathbb{G}_q$, and a and b are chosen at random in $\mathbb{Z}_q$.

Theorem 2. *The proposed scheme is an (m, T)-indistinguishable scheme as described in Def. 3 assuming the difficulty of the DDH problem in $\mathbb{G}_q$.*

Proof. Here, assuming that there exists a probabilistic polynomial time adversary A which can break our proposed scheme, we show that it is possible to construct another adversary B which can solve the DDH problem with a non-negligible advantage.

For a input (g_1, g_2, h_1, h_2), B solves the DDH problem as follows. First, B chooses $x_{i,j}, y_{i,j} \in \mathbb{Z}_q$ for $i = 0, \cdots, 2k-1; j = 0, \cdots, m$ and calculates a public key $PK := (g_1, g_2, p, q, g_1^{x_{0,0}} g_2^{y_{0,0}}, \cdots, g_1^{x_{2k-1,m}} g_2^{y_{2k-1,m}})$. Next, B gives PK to A, and A submits queries to key exposure oracle according to the restrictions in Def. 1 and 3. When A submits a query t_p to $Exp_t(\cdot)$, it returns correct secret keys $\{SK_{u_i,t_p}\}_{i=1,\cdots,N}$. Since $Exp_t(\cdot)$ knows all of the coefficients, it can return secret keys correctly. Then, A submits a query (t^*, M_0, M_1) to the left-or-right encryption oracle. B, then, picks $b \in_R \{0, 1\}$ and returns a challenge ciphertext $c := (h_1, h_2, z_{t^*,0}, z_{t^*,1}, \cdots, z_{t^*,2k-1})$ such that $z_{t^*,0} = M_b \prod_{j=0}^{m} (h_1^{x_{0,j}} h_2^{y_{0,j}})^{t^j}$, $z_{t^*,i} = \prod_{j=0}^{m} (h_1^{x_{i,j}} h_2^{y_{i,j}})^{t^j}$ for $i = 1, \cdots, 2k-1$.

It is clear that if (g_1, g_2, h_1, h_2) is a DDH-tuple, then c^* is a valid ciphertext of M_b. On the other hand, if it is a random tuple, it is information theoretically impossible to obtain any information on b (due to randomness of "$\log_{h_1} h_2$"). Letting b' be A's output, B outputs "(g_1, g_2, h_1, h_2) is a DDH-tuple" if $b' = b$, otherwise, B outputs "(g_1, g_2, h_1, h_2) is a random tuple". Consequently, B solves the DDH problem with a non-negligible advantage. □

5 Application

When users receive content distribution service at home, they store their secret keys in their security devices installed in their STBs and use their secret keys to decrypt contents. Current digital broadcasting systems use a smart card as a PSD; it is called Conditional Access System (CAS) card [1]. The CAS cards are now used to protect broadcasting content copyright by more than 11,200,000 subscribers (users) in Japan. Subscribers are able to receive its service only at home, because they cannot extract their secret keys from their CAS cards.

If it were possible to copy their secret keys, subscribers would be able to obtain services beyond the proximity of the STB. It is "Anywhere TV service". The system could be constructed easily, if broadcasters (content providers) allowed subscribers to copy the secret keys to PMDs, but there could be a problem for broadcasters to be exposed to serious damage. They are unwilling to accept such risk, so they do not allow their subscribers to copy their secret keys.

The system, which enables subscribers to take along their secret keys and minimizes damage of subscriber's key exposure, is required. The TTaKE scheme, that we proposed, is useful for such a service. It is compatible with the current broadcasting system and it meets both of the broadcasters' and subscribers' requirements.

5.1 TTaKE Application to Anywhere TV Service

As an example of anywhere TV service, we show that a TTaKE scheme can be used for a broadcasting service which contains series type drama program. Each subscriber has a STB at his house, and the STB contains a CAS card.

A broadcaster sets the total number of episodes of series type drama program T, and distributes each episode week by week in turn. Each subscriber get a master key SK_u^*, and an initial key $SK_{u,0}$ prior to the drama series start. SK_u^* is stored in the subscriber's CAS card. The card can force him not to copy SK_u^*. $SK_{u,0}$ is stored into storage of his STB. $SK_{u,0}$ is only used to create the 1st secret key $SK_{u,1}$.

When a broadcaster distributes $j(j = 1, \cdots, T)$th episode, it encrypts content with the episode number j and the public key. Each subscriber has to create jth secret key $SK_{u,j}$. When he holds $SK_{u,j-1}$ in his PMD, he sets his PMD to his STB. The STB inputs the index j to the CAS card and the CAS card returns jth partial key $SK_{u,j}^{'}$. Simultaneously, the STB gets $SK_{u,j-1}$ from the PMD/storage of STB, and erases $SK_{u,j-1}$ from there. The STB then calculates $SK_{u,j}$ with $SK_{u,j-1}$ and $SK_{u,j}^{'}$, and then erases $SK_{u,j-1}$ and $SK_{u,j}^{'}$ it holds. When he travels somewhere and wants to watch the jth episode there, he sets the PMD to another STB, and can watch the jth episode.

Although malicious use of secret keys and key exposure have to been taken into accounts, owing to TTaKE it can be secure against such attacks.

5.2 System Structure

We then show system structures. Figure 1 shows the current broadcasting system standardized in Japan [1] and anywhere TV system we propose. *Scramble* is content encryption process, which uses symmetric encryption scheme with a secret key Ks. Ks is decrypted in each CAS card. Km is a unique key of each CAS card and it is used to transmit individual data to each CAS card. These processes are common in the two systems and the anywhere TV system can have compatibility with the current broadcasting system.

Kw is a unique key for each contract in the current system. It is encrypted with Km and transmitted to each CAS card after a user makes a contract with

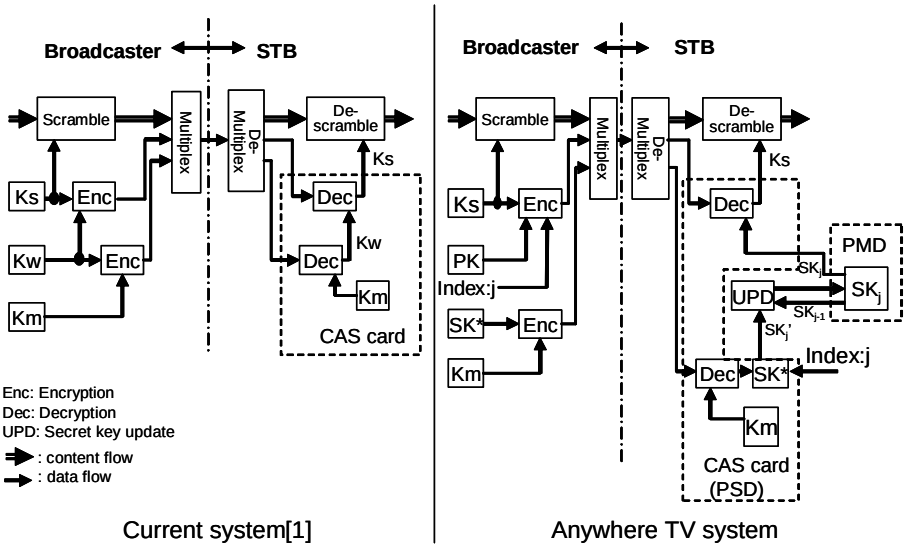


Fig. 1. Comparison of Current Broadcasting System [1] and Anywhere TV system

a broadcaster. Kw is a key for symmetric encryption and unchanged during all service periods, so Kw's exposure affects all episodes copyrights. In addition, only the providers, which know Kw, can use this system.

In the anywhere TV system, SK^* and SK_0 (only SK^* is described in Fig. 1) are encrypted and transmitted to each CAS card. Instead of Kw, SK^* is transmitted after a contract, and Km is used to encrypt this individual data. This process is compatible with that of current broadcasting system. Each decryption key, SK_j, is different from the other time periods', so the key's exposure affects only the jth episode. Moreover, this system employs a public key cryptosystem. Although CPU cost becomes high, any broadcasters can use this system and the system improves content usability.

As described above, a TTaKE scheme could apply to the anywhere TV service, and the system has compatibility with the current broadcasting system.

References

1. "Conditional Access System Specifications for Digital Broadcasting ARIB-STD-B25," Association of Radio Industries and Businesses.
2. D. Boneh and M. Franklin, "An Efficient Public Key Traitor Tracing Scheme," Proc. of Crypto'99, LNCS 1666, Springer-Verlag, pp.338-353, 1999.
3. O. Billet and H. Gilbert, "A Traceable Block Cipher," Proc. of Asiacrypt'03, LNCS 2894, Springer-Verlag, pp.331-346, 2003.
4. M. Bellare and S. K. Miner, "A Forward-Secure Digital Signature Scheme," Proc. of Crypto'99, LNCS 1666, Springer-Verlag, pp.431-448, 1999.
5. B. Chor, A. Fiat and M. Naor, "Tracing Traitors," Proc. of Crypto'94, LNCS 839, Springer-Verlag, pp.257-270, 1994.

6. Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung, "A Generic Construction for Intrusion-Resilient Public-Key Encryption," Proc. of CT-RSA'04, LNCS 2964, Springer-Verlag, pp.81-98, 2004.
7. Y. Dodis, J. Katz, S. Xu and M. Yung, "Key-Insulated Public-Key Cryptosystems," Proc. of Eurocrypt'02, LNCS 2332, Springer-Verlag, pp.65-82, 2002.
8. K. Kurosawa and Y. Desmedt, "Optimum Traitor Tracing and Asymmetric Schemes," Proc. of Eurocrypt'98, LNCS 1403, Springer-Verlag, pp.145-157, 1998.
9. K. Kurosawa and T. Yoshida, "Linear Code Implies Public-Key Traitor Tracing," Proc. of PKC'02, LNCS 2274, Springer-Verlag, pp.172-187, 2002.
10. T. Matsushita and H. Imai, "A Public-Key Black-Box Traitor Tracing Scheme with Sublinear Ciphertext Size Against Self-Defensive Pirates," Proc. of Asiacrypt'04, LNCS 3329, Springer-Verlag, pp.260-275, 2004.
11. D. R. Stinson and R. Wei, "Key Preassigned Traceability Schemes for Broadcast Encryption," Proc. of Selected Areas in Cryptography (SAC'98), LNCS 1556, Springer-Verlag, pp.144-156, 1998.
12. V. D. To, R. Safavi-Naini, and F. Zhang, "New traitor tracing schemes using bilinear map," Proc. of ACM Workshop on Digital Rights Management (DRM'03), pp.67-76, 2003.

Fingercasting—Joint Fingerprinting and Decryption of Broadcast Messages

André Adelsbach, Ulrich Huber, and Ahmad-Reza Sadeghi

Horst Görtz Institute for IT Security
Ruhr-Universität Bochum, Germany
andre.adelsbach@nds.rub.de, {huber, sadeghi}@crypto.rub.de

Abstract. We propose a stream cipher that provides confidentiality, traceability and renewability in the context of broadcast encryption. We prove it to be as secure as the generic pseudo-random sequence on which it operates. This encryption scheme, termed fingercasting scheme, achieves joint decryption and fingerprinting of broadcast messages in such a way that an adversary cannot separate both operations or prevent them from happening simultaneously. The scheme is a combination of a broadcast encryption scheme, a fingerprinting scheme and an encryption scheme inspired by the Chameleon cipher. It is the first to provide a formal security proof and a non-constant lower bound for resistance against collusion of malicious users, i.e., a minimum number of content copies needed to remove all fingerprints. The scheme is efficient and includes parameters that allow, for example, to trade-off storage size for computation cost at the receiving end.

1 Introduction

Experience shows that adversaries attack Broadcast Encryption (BE) systems in a variety of different ways. Their attacks may be on the hardware that stores cryptographic keys, e.g., when they extract keys from a compliant device to develop a pirate device such as the DeCSS software that circumvents the Content Scrambling System [1]. Alternatively, their attacks may be on the decrypted content, e.g., when a legitimate user shares decrypted content with illegitimate users on a file sharing system such as Napster, Kazaa, and BitTorrent.

The broadcasting sender thus has three security requirements: *confidentiality*, *traceability* of content and keys, and *renewability* of the encryption scheme. Confidentiality tries to prevent illegal copies, whereas traceability is a second line of defense aimed at finding the origin of an illegal copy. The need for traceability implies that confidentiality may be compromised in rare cases, e.g., when a few users illegally distribute their secret keys. Renewability ensures that after such rare events, the encryption system can recover from the security breach.

In broadcasting systems deployed today, e.g., CPPM [2] or AACS [3], confidentiality and renewability often rely on BE because it provides short ciphertexts while at the same time having realistic storage requirements in devices and acceptable computational overhead. Traitor tracing enables traceability of keys,

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 136–147, 2006.
© Springer-Verlag Berlin Heidelberg 2006

whereas fingerprinting provides traceability of content. Finally, renewability may be achieved using revocation of the leaked keys.

However, none of the mentioned cryptographic schemes covers all three security requirements. Some BE schemes lack traceability of keys, whereas no practically relevant scheme provides traceability of content [4, 5, 6, 7]. Traitor tracing does not provide traceability of of content [8, 9]. Fingerprinting schemes do not provide confidentiality [10]. The original Chameleon cipher provides confidentiality, traceability and a hint on renewability, but with a small constant bound for collusion resistance and without formal proof of security [11]. Asymmetric schemes, which provide each compliant device with a certificate and accompany content with Certificate Revocation Lists (CRLs), lack traceability of content and reach the limits of renewability when CRLs become too large. A trivial combination of fingerprinting and encryption leads to an unacceptable transmission overhead because the sender needs to sequentially transmit each fingerprinted copy. Finally, receiver-side fingerprint embedding relies on the tamper resistance of the receivers' hardware [10], which is doubtful in practice. Each receiver is trusted to embed the fingerprint *after* decryption. However, perfect tamper-resistance cannot be achieved under realistic assumptions [15].

We present, to the best of our knowledge, the first rigorous security proof of Chameleon ciphers, thus providing a sound foundation for their recent applications, e.g., [12]. Furthermore, we give an explicit criterion to judge the security of the Chameleon cipher's key table. Our *fingercasting* approach fulfills all three security requirements at the same time. It is a combination of (i) a new Chameleon cipher based on the *finger*printing capabilities of a class of watermarking schemes and (ii) an arbitrary broad*cast* encryption scheme, which explains the name of the approach. The basic idea is to use the Chameleon cipher for combining decryption and fingerprinting. To achieve renewability, we use a BE scheme to provide fresh session keys as input to the Chameleon cipher. To achieve traceability, we fingerprint the receivers' key tables such that they embed a fingerprint into the content during decryption. To enable higher collusion resistance than the original Chameleon scheme, we tailor our cipher to emulate any watermarking scheme whose coefficients can be disaggregated into additive components. As proof of concept, we instantiate the watermarking scheme with Spread Spectrum Watermarking (SSW), which has proven collusion resistance [13, 14]. However, we might as well use any other such scheme.

We note that our fingercasting approach distributes a single encrypted copy of the content. In addition, it ensures embedding of a fingerprint even if a malicious user succeeds in extracting the decryption keys of his receiver. As long as the number of colluding users remains below a threshold, they can only create new decryption keys and content copies that incriminate at least one of them.

2 Related Work

The original Chameleon cipher of Anderson and Manifavas is a 3-collusion-resistant fingercasting scheme [11]: A collusion of up to 3 malicious users has a negligible chance of creating of a good copy that does not incriminate them.

Each legitimate user knows the seed of a Pseudo-Random Sequence (PRS) and a long table filled with random keywords. Based on the sender's master table, each receiver obtains a slightly different table copy, where individual bits are modified in a characteristic way. Interpreting the PRS as a sequence of addresses in the table, the sender adds the corresponding keywords in the master table bitwise modulo 2 in order to mask the plaintext word. The receiver applies the same operation to the ciphertext using its table copy, thus embedding the fingerprint.

The original cipher, however, has some inconveniences. Most importantly, it has no formal security analysis and bounds the collusion resistance by the constant number 3, whereas our scheme allows to choose this bound depending on the number of available watermark coefficients. In addition, the original scheme limits the content space (and keywords) to strings with characteristic bit positions that may be modified without visibly altering the content. In contrast, our scheme uses algebraic operations in a group of large order, which enables modification of any bit in the keyword and processing of arbitrary documents.

Chameleon was inspired by work from Maurer [16]. His cipher achieves information-theoretical security in the bounded storage model with high probability. In contrast, Chameleon and our proposed scheme only achieve computational security. However, Maurer's cipher was never intended to provide traceability of content or renewability, but only confidentiality.

Ferguson et al. discovered security weaknesses in a randomized stream cipher similar to Chameleon [17]. However, their attack only works for linear sequences of keywords in the master table, not for the PRSs of our proposed solution.

Ergun, Kilian, and Kumar prove that an averaging attack with additional Gaussian noise defeats any watermarking scheme [18]. Their bound on the minimum number of different content copies needed for the attack asymptotically coincides with the bound on the maximum number of different content copies to which the watermarking scheme of Kilian et al. is collusion-resistant [14]. As we emulate [14], its collusion resistance is asymptotically the best we can hope for.

Recently there was a great deal of interest in joint fingerprinting and decryption [12, 19, 20, 10, 21]. Basically, we can distinguish three strands of work. The first strand of work applies Chameleon in different application settings. Briscoe et al. introduce Nark, which is an application of the original Chameleon scheme in the context of Internet multicast [12]. However, in contrast to our new Chameleon scheme they neither enhance the original scheme nor analyze its security. The second strand of work tries to achieve joint fingerprinting and decryption by either trusting network nodes to embed fingerprints (Watercasting in [19]) or doubling the size of the ciphertext by sending differently fingerprinted packets of content [20]. Our proposed solution neither relies on trusted network nodes nor increases the ciphertext size. The third strand of work proposes new joint fingerprinting and decryption processes, but at the price of replacing encryption with scrambling, which does not achieve indistinguishability of ciphertext and has security concerns [10, 21]. In contrast, our new Chameleon scheme achieves indistinguishability of ciphertext.

3 Preliminaries

3.1 Notation

We recall some standard notations that will be used throughout the paper. First, we denote scalar objects with lower-case variables, e.g., o_1, and object tuples as well as roles with upper-case variables, e.g., X_1. When we summarize objects or roles in set notation, we use an upper-case calligraphic variable, e.g., $\mathcal{O} := \{o_1, o_2, \ldots\}$ or $\mathcal{X} := \{X_1, X_2, \ldots\}$. Second, let A be an algorithm. By $y \leftarrow \mathsf{A}(x)$ we denote that y was obtained by running A on input x. For example, by $y \leftarrow \mathsf{N}(\mu, \sigma)$ we denote that y was obtained by selecting it at random with normal distribution, where μ is the mean and σ the standard deviation. Third, $o_1 \xleftarrow{R} \mathcal{O}$ and $o_2 \xleftarrow{R} [0, z]$ denote the selection of a random element of the set $\mathcal{O}$ and the interval $[0, z]$ with uniform distribution. Finally, $V \cdot W$ denotes the dot product of two vectors $V := (v_1, \ldots, v_n)$ and $W := (w_1, \ldots, w_n)$, which is defined as $V \cdot W := \sum_{j=1}^{n} v_j w_j$, while $||V||$ denotes the Euclidean norm $||V|| := \sqrt{V \cdot V}$.

3.2 Roles and Objects in Our System Model

The *(broadcast) center* manages the broadcast channel, distributes decryption keys and is fully trusted. The *users* obtain the content via devices that we refer to as *receivers*. For example, a receiver may be a set-top box in the context of pay-TV or a DVD player in movie distribution. We denote the number of receivers with N; the set of receivers is $\mathcal{U} := \{u_i \mid 1 \leq i \leq N\}$. When a receiver violates the terms and conditions of the application, e.g., leaks its keys or shares content, the center revokes the receiver's keys and thus makes them useless for decryption purposes. We denote the set of revoked receivers with $\mathcal{R} := \{r_1, r_2, \ldots\} \subset \mathcal{U}$.

We represent broadcast content as a sequence $M := (m_1, \ldots, m_n)$ of real numbers in $[0, z]$, where M is an element of the content space $\mathcal{M}$. For example, these numbers may be the n most significant coefficients of the Discrete Cosine Transform (DCT) of an image as described in [13]. However, they should not be thought of as a literal description of the underlying content, but as a representation of the values that are to be changed by the watermarking process [18].

3.3 Cryptographic Building Blocks

Chameleon Encryption. To set up the scheme $\mathcal{CE} := (\mathsf{KeyGenCE}, \mathsf{KeyExtrCE}, \mathsf{EncCE}, \mathsf{DecCE})$, the center generates the secret master table MT and the secret table fingerprints $TF := (TF^{(1)}, \ldots, TF^{(N)})$ using the key generation algorithm $(MT, TF) \leftarrow \mathsf{KeyGenCE}(N, 1^{\lambda'}, par_{\mathsf{CE}})$, where N is the number of receivers, λ' a security parameter, and par_{CE} a set of parameters. To add receiver u_i to the system, the center uses the key extraction algorithm $RT^{(i)} \leftarrow \mathsf{KeyExtrCE}(MT, TF, i)$ to deliver the secret receiver table $RT^{(i)}$ to u_i. To encrypt content M exclusively for the receivers in possession of a receiver table $RT^{(i)}$ and a session key k^{sess}, the center uses the encryption algorithm $C \leftarrow \mathsf{EncCE}(MT, k^{\mathsf{sess}}, M)$, where the output is the ciphertext C. Only a receiver u_i in possession of $RT^{(i)}$

and k^{sess} is capable of decrypting C and obtaining a fingerprinted copy $M^{(i)}$ of M using the decryption algorithm $M^{(i)} \leftarrow \mathsf{DecCE}(RT^{(i)}, k^{\mathsf{sess}}, C)$. When the center discovers an illegal copy M^* of content M, it uses the fingerprint detection algorithm of the underlying fingerprinting scheme.

Fingerprinting. To set up the scheme, the center generates the secret content fingerprints $CF := (CF^{(1)}, \ldots, CF^{(N)})$ and the secret similarity threshold t using the setup algorithm $(CF, t) \leftarrow \mathsf{SetupFP}(N, n', par_{\mathsf{FP}})$, where N is the number of receivers, n' the number of content coefficients, and par_{FP} a set of performance parameters. To embed the content fingerprint $CF^{(i)} := (cf_1^{(i)}, \ldots, cf_{n'}^{(i)})$ of receiver u_i into the original content M, the center uses the embedding algorithm $M^{(i)} \leftarrow \mathsf{EmbedFP}(M, CF^{(i)})$. To verify whether an illegal copy M^* of content M contains traces of the content fingerprint $CF^{(i)}$ of receiver u_i, the center uses the detection algorithm $dec \leftarrow \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)$. It calculates the similarity between the detected fingerprint $CF^* := M^* - M$ and $CF^{(i)}$ using a similarity measure. If the similarity is above the threshold t, then the center declares u_i guilty ($dec = \mathtt{true}$), otherwise innocent ($dec = \mathtt{false}$). The detection algorithm is called non-blind because it needs the original content M as input.

In the sequel, we call a fingerprinting scheme *additive* if the probability distribution Prob of its coefficients has the following property: Adding two independent random variables that follow Prob results in a random variable that also follows Prob. Spread Spectrum Watermarking (SSW) is an instance of an additive fingerprinting scheme [14]. The content fingerprint $CF^{(i)}$ consists of independent random variables $cf_j^{(i)}$ with distribution $\mathsf{Prob} = \mathsf{N}(0, \sigma')$, where σ' is a function $\mathsf{f}_{\sigma'}(N, n', par_{\mathsf{FP}})$. The similarity threshold t is a function $\mathsf{f}_t(\sigma', N, par_{\mathsf{FP}})$. Both functions $\mathsf{f}_{\sigma'}$ and f_t are specified in [14]. During $\mathsf{EmbedFP}$, the center adds the fingerprint coefficients to the content coefficients: $m_j^{(i)} \leftarrow m_j + cf_j^{(i)}$. The similarity measure is $\mathsf{Sim}(CF^*, CF^{(i)}) := (CF^* \cdot CF^{(i)})/||CF^*||$.

Theorem 1. [14, Section 3.4] *In the SSW scheme with the above parameters, an adversarial coalition needs $\Omega(\sqrt{n'/\ln N})$ differently fingerprinted copies of content M to have a non-negligible chance of creating a good copy M^* without any coalition member's fingerprint.*

Broadcast Encryption. To set up the scheme, the center generates the secret master key MK using the key generation algorithm $MK \leftarrow \mathsf{KeyGenBE}(N, 1^{\lambda''})$, where N is the number of receivers and $1^{\lambda''}$ the security parameter. To add receiver u_i to the system, the center uses the key extraction algorithm $SK^{(i)} \leftarrow \mathsf{KeyExtrBE}(MK, i)$ to extract the secret key $SK^{(i)}$ of u_i. To encrypt content M exclusively for the non-revoked receivers $\mathcal{U} \setminus \mathcal{R}$, the center uses the encryption algorithm $C \leftarrow \mathsf{EncBE}(MK, \mathcal{R}, M)$, where the output is the ciphertext C. Only a non-revoked receiver u_i has a matching private key $SK^{(i)}$ that allows to decrypt C and obtain M using the decryption algorithm $M \leftarrow \mathsf{DecBE}(i, SK^{(i)}, C)$.

Pseudo-random Sequence (PRS). We formally define the term PRS in the technical report [22]. Informally, a PRS is a long bit string that no efficient algorithm can distinguish from a truly random bit string of identical length. In

order to create a PRS, a Pseudo-Random Sequence Generator (PRSG), which is a deterministic polynomial-time algorithm G, derives the long bit string from a short random seed. If $|\sigma| \in \mathbb{N}$ is the length of the seed σ, then the expansion factor $\mathsf{len} : \mathbb{N} \to \mathbb{N}$ of the PRSG determines the length of the PRS: $|\mathsf{G}(\sigma)| = \mathsf{len}(|\sigma|)$.

3.4 Requirements of a Fingercasting Scheme

Before we enter into the details of our fingercasting (FC) approach, we summarize its requirements: correctness, security, collusion resistance, and frame-proofness. To put it simply, the aim of the fingercasting approach is to generically combine an instance of a BE scheme, a Chameleon scheme, and a fingerprinting scheme such that the combination inherits the security of BE and Chameleon scheme as well as the collusion resistance of fingerprinting.

We informally explain the requirements one by one; a formal definition is available in [22]. Correctness requires that the receiver obtains with high probability a fingerprinted copy that is perceptually indistinguishable from the original; in other words, the fingerprint may not deteriorate the content and lead to a bad copy. We denote the residual probability of a bad copy with p^{bad}, which obviously should be close to 0 for practical purposes. The SSW scheme of [14] uses the measure $||M^{(i)} - M|| \leq \sqrt{n'}\delta$ to decide whether a copy is good, where n' is the number of content coefficients and δ a goodness criterion.

All relevant BE schemes provide even stronger security notions than Chameleon [5, 6, 7]. The remaining requirements therefore only relate to the Chameleon scheme $\mathcal{CE}$. We informally define security of $\mathcal{CE}$ as follows: No efficient algorithm may be capable of distinguishing the ciphertexts of two messages, even if this algorithm selects the two messages to be encrypted and obtains an encryption of one of the two messages selected at random.

Resistance against a collusion of up to q colluders (or q-collusion resistance) means that no efficient algorithm with access to the secret information of at most q colluders may be capable of creating a good content copy such that the fingerprint detection algorithm incriminates none of the colluders. We denote the residual probability of a successful collusion with p^{neg}, which is usually called the false negative probability. 1-collusion resistance is known as robustness.

Frame-proofness is similar to collusion resistance, but the adversarial coalition wins if the detection algorithm accuses an innocent user. We denote the residual probability of a successful framing attack with p^{pos}, which is usually called the false positive probability.

4 Proposed Solution

4.1 High-Level Overview of the Proposed Fingercasting Scheme

To fingercast content, the center uses the BE scheme to send a fresh session key to each non-revoked receiver. This session key initializes a pseudo-random sequence generator. The resulting pseudo-random sequence represents a sequence

of addresses in the master table of our new Chameleon scheme. The center encrypts the content with the master table entries to which the addresses refer. Each receiver has a unique receiver table that differs only slightly from the master table. During decryption, these slight differences in the receiver table lead to slight, but characteristic differences in the content copy.

We divide this approach into the same five steps that we have seen for Chameleon schemes in Section 3.3. First, the *key generation* algorithm of the fingercasting scheme consists of the key generations algorithms of the two underlying schemes $\mathsf{KeyGenBE}$ and $\mathsf{KeyGenCE}$. The center's master key thus consists of MK, MT and TF. Second, the same observation holds for the *key extraction* algorithm of the fingercasting scheme. It consists of the respective algorithms in the two underlying schemes $\mathsf{KeyExtrBE}$ and $\mathsf{KeyExtrCE}$. The secret key of receiver u_i therefore has two elements: $SK^{(i)}$ and $RT^{(i)}$.

Third, the *encryption* algorithm defines how we interlock the two underlying schemes. To encrypt, the center generates a fresh and random session key $k^{\mathsf{sess}} \xleftarrow{R} \{0,1\}^{\lambda}$. This session key is broadcasted to the non-revoked receivers using the BE scheme: $C_{\mathsf{BE}} \leftarrow \mathsf{EncBE}(MK, \mathcal{R}, k^{\mathsf{sess}})$. Subsequently, the center uses k^{sess} to determine addresses in the master table MT of the Chameleon scheme and encrypts with the corresponding entries: $C_{\mathsf{CE}} \leftarrow \mathsf{EncCE}(MT, k^{\mathsf{sess}}, M)$. The ciphertext of the fingercasting scheme thus has two elements C_{BE} and C_{CE}.

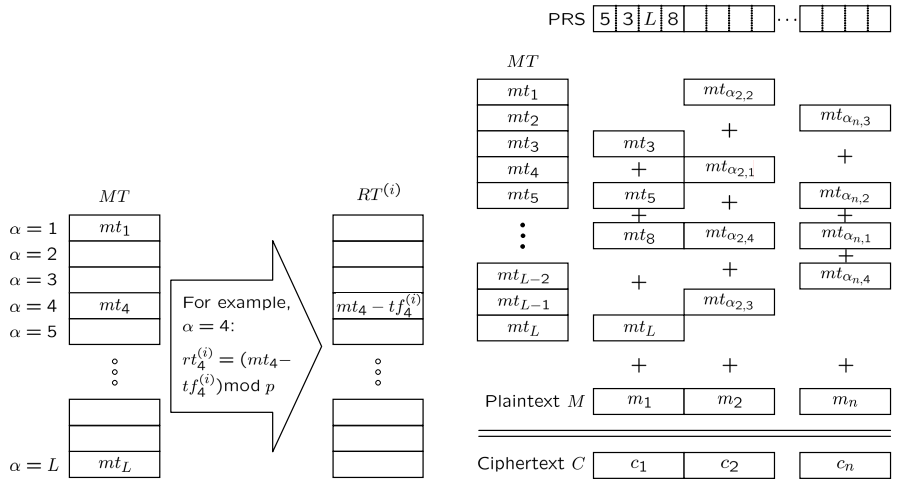
Fourth, the *decryption* algorithm inverts the encryption algorithm with unnoticeable, but characteristic errors. First of all, each non-revoked receiver u_i recovers the correct session key: $k^{\mathsf{sess}} \leftarrow \mathsf{DecBE}(i, SK^{(i)}, C_{\mathsf{BE}})$. Therefore, u_i can recalculate the PRS and the correct addresses in receiver table $RT^{(i)}$. However, this receiver table is slightly different from the master table. Therefore, u_i obtains a fingerprinted copy $M^{(i)}$ that is slightly different from the original content: $M^{(i)} \leftarrow \mathsf{DecCE}(RT^{(i)}, k^{\mathsf{sess}}, C_{\mathsf{CE}})$. Last, the *fingerprint detection* algorithm of the fingercasting scheme is identical to that of the underlying fingerprinting scheme.

4.2 A New Chameleon Scheme

Up to now, we have focused on the straightforward aspects of our approach; we have neglected the impact of the requirements on the Chameleon scheme. In the sequel, we will show a specific Chameleon scheme that fulfills all of them. We design it such that its content fingerprints can emulate any additive fingerprinting scheme, which we later instantiate with the SSW scheme as proof of concept.

Key Generation. To define this algorithm, we need to determine how the center generates the master table MT and the table fingerprints TF. To generate MT, the center chooses $L = 2^l$ table entries at random from the interval $[0, z]$ with independent uniform distribution: $mt_\alpha \xleftarrow{R} [0, z]$ for all $\alpha \in \{1, \ldots, L\}$. The center thus obtains the master table $MT := (mt_1, mt_2, \ldots, mt_L)$.

To generate the table fingerprints $TF := (TF^{(1)}, \ldots, TF^{(N)})$, the center selects for each receiver u_i and each master table entry mt_α a fingerprint coefficient in order to disturb the original entry. Specifically, each fingerprint coefficient $tf_\alpha^{(i)}$



(a) To derive $RT^{(i)}$ from MT, the center subtracts the L fingerprint coefficients $tf_\alpha^{(i)}$ at address α for all $\alpha \in \{1, \ldots, L\}$.

(b) To derive C from M, the center uses the session key to generate a PRS. It adds the addressed master table entries to M.

Fig. 1. Receiver table derivation and ciphertext calculation

of table fingerprint $TF^{(i)}$ is independently distributed according to the probability distribution Prob of the additive fingerprinting scheme, but scaled down with an attenuation factor $f \in \mathbb{R}$, $f \geq 1$:

$$tf_\alpha^{(i)} \leftarrow 1/f \cdot \mathsf{Prob}(par_{\mathsf{FP}}) \tag{1}$$

Key Extraction. After the probabilistic key generation algorithm we now describe the deterministic key extraction algorithm. The center processes table fingerprint $TF^{(i)} := (tf_1^{(i)}, \ldots, tf_L^{(i)})$ of receiver u_i as follows: The center subtracts each fingerprint coefficient in $TF^{(i)}$ from the corresponding master table entry to obtain the receiver table entry, which we illustrate in Fig. 1(a):

$$\forall\, \alpha \in \{1, \ldots, L\} : \quad rt_\alpha^{(i)} \leftarrow mt_\alpha - tf_\alpha^{(i)} \bmod p \tag{2}$$

Remark 1. The modulo operator allows only integer values to be added. However, MT, TF, and M are based on real numbers. We solve this ostensible contradiction by scaling the real values to the integer domain with an appropriate scaling factor ρ and ignoring further decimal digits. ρ must be large enough to allow a computation in the integer domain with a sufficiently high precision suitable for the current application. We implicitly assume this scaling to the integer domain whenever real values are used. For example, with real-valued variables $rt^{(i)}$, mt, and $tf^{(i)}$ the operation $rt^{(i)} \leftarrow (mt - tf^{(i)}) \bmod p$ actually stands for $\rho \cdot rt^{(i)} \leftarrow (\rho \cdot mt - \rho \cdot tf^{(i)}) \bmod p$. Note that the scaling operation does not give the adversary more information than in the original fingerprinting scheme.

Encryption. Fig. 1(b) gives an overview of the encryption algorithm. The session key k^{sess} is used as the seed of a PRSG with $\mathsf{len}(|k^{\mathsf{sess}}|) \geq n \cdot s \cdot l$, where s will be specified below. To give a practical example for a PRSG, k^{sess} may serve as the key for a conventional block cipher, e.g., AES, in output feedback mode. Each block of l bits of the PRS is interpreted as an address β in the master table MT. For each coefficient of the plaintext, the center uses s addresses that define s entries of the master table. In total, the center obtains $n \cdot s$ addresses that we denote with $\beta_{j,k}$, where j is the coefficient index, k the address index, and $\mathsf{Extract}_i$ extracts the i-th block of length l from its input string:

$$\forall\, j \in \{1, \ldots, n\},\ \forall\, k \in \{1, \ldots, s\} : \quad \beta_{j,k} \leftarrow \mathsf{Extract}_{(j-1)s+k}(\mathsf{G}(k^{\mathsf{sess}})) \tag{3}$$

For each content coefficient, the center adds the s master table entries modulo the group order. In Fig. 1(b), we illustrate the case $s = 4$, which is the design choice in the original Chameleon cipher. Let $mt_{\beta_{j,k}}$ be the master table entry referenced by address $\beta_{j,k}$ from (3). Then coefficient c_j of C is calculated as

$$c_j \leftarrow \left(m_j + \sum_{k=1}^{s} mt_{\beta_{j,k}} \right) \bmod p\ , \quad j \in \{1, \ldots, n\}\ . \tag{4}$$

Decryption. The decryption algorithm proceeds in the same way as the encryption algorithm with two exceptions. First, the receiver has to use its receiver table $RT^{(i)}$ instead of MT. Second, the addition is replaced by subtraction. The j-th coefficient $m_j^{(i)}$ of the plaintext copy $M^{(i)}$ of receiver u_i is thus calculated as

$$m_j^{(i)} \leftarrow \left(c_j - \sum_{k=1}^{s} rt_{\beta_{j,k}}^{(i)} \right) \bmod p, \tag{5}$$

where $rt_{\beta_{j,k}}^{(i)}$ denotes the receiver table entry of receiver u_i referenced by address $\beta_{j,k}$ generated in (3). As the receiver table $RT^{(i)}$ slightly differs from MT, the plaintext copy $M^{(i)}$ obtained by receiver u_i slightly differs from M. The ratio of distortion is the same as that of the instantiated fingerprinting scheme.

Fingerprint Detection. When the center detects an illegal copy $M^*{=}(m_1^*, \ldots, m_n^*)$ of M, it tries to identify the receivers that participated in the generation of M^*. To do so, the center verifies whether the fingerprint of a suspect receiver u_i is present in M^*. Obviously, the fingerprint is unlikely to appear in its original form; an adversary may have modified it by applying common attacks such as resampling, requantization, and compression. In addition, an adversarial coalition may have colluded and created M^* using several different copies of M.

The fingerprint detection algorithm is identical to that of the underlying fingerprinting scheme: $dec \leftarrow \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)$. In order to scale the content fingerprint, we need to select the attenuation factor f in (1). We choose it such that adding s attenuated fingerprint coefficients generates a random variable that follows Prob *without* attenuation. We give an example in Section 4.3.

In order to verify whether the table fingerprint $TF^{(i)}$ of receiver u_i left traces in M^*, $\mathsf{DetectFP}$ calculates the similarity between the detected content fingerprint

CF^* with coefficients $cf_j^* := m_j^* - m_j$ and the content fingerprint $CF^{(i)}$ in u_i's copy $M^{(i)}$ with $cf_j^{(i)} := m_j^{(i)} - m_j \overset{(4),(5)}{=} \sum_{k=1}^{s}(mt_{\beta_{j,k}} - rt_{\beta_{j,k}}^{(i)}) \overset{(2)}{=} \sum_{k=1}^{s} tf_{\beta_{j,k}}^{(i)}$, where $tf_{\beta_{j,k}}^{(i)}$ is the fingerprint coefficient that fingerprinted receiver table $RT^{(i)}$ at address $\alpha = \beta_{j,k}$ in (2). If the similarity is above threshold t, the center declares u_i guilty. Note that the calculation of CF^* necessitates the original content M, whereas the calculation of $CF^{(i)}$ relies on the session key k^{sess} and the table fingerprint $TF^{(i)}$; the scheme is thus non-blind. The same algorithm applies to detection of fingerprints in illegal copies of receiver tables. Their fingerprints have the same construction and statistical properties (for further details see [22]).

Parameter Selection. The scheme has two major parameters L and s that allow a trade-off between the size of $RT^{(i)}$, which u_i has to store, and the computation cost, which grows linearly with the number s of addresses per content coefficient in (4). By increasing L, we can decrease s in order to replace computation cost with storage size. Further details appear in [22].

4.3 Instantiation with Spread Spectrum Watermarking

In this section, we instantiate the fingerprinting scheme with the SSW scheme of [14] and thereby inherit its collusion resistance and frame-proofness. Let the center choose the SSW scheme's parameters $par_{\mathsf{FP}} = (\delta, p^{\mathsf{bad}}, p^{\mathsf{pos}})$, which allows to calculate a standard deviation σ' and a threshold t via two functions $\mathsf{f}_{\sigma'}(N, n', \delta, p^{\mathsf{bad}})$ and $\mathsf{f}_t(\sigma', N, p^{\mathsf{pos}})$ defined in [14]. The probability distribution of the SSW scheme is then $\mathsf{Prob} = \mathsf{N}(0, \sigma')$. We set $f = s$ such that $1/f \cdot \mathsf{N}(0, \sigma')$ in (1) is still a normal distribution with mean 0 and standard deviation $1/\sqrt{s} \cdot \sigma'$. It remains to define the similarity measure for the detection algorithm $dec \leftarrow \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)$, which [14] defines as $dec = \mathtt{true}$ if $(CF^* \cdot CF^{(i)})/||CF^*|| > t$. We call this instantiation *exact* if it achieves the same statistical properties as the fingerprinting scheme that it instantiates. Theorem 2 below states that the above choice is an exact instantiation; we prove it in [22]:

Theorem 2. *Let σ' and σ be the standard deviations of the SSW scheme and the Chameleon scheme instantiated with SSW, respectively, and n' and n be their number of content coefficients. Then the following mapping between both schemes is an exact instantiation: $\sigma' = \sqrt{s} \cdot \sigma$ $(\Leftrightarrow f = s)$ and $n' = n$*

4.4 Analysis

Correctness, Collusion Resistance and Frame-Proofness. Correctness follows trivially from the correctness of the two underlying schemes, i.e., the BE scheme and the SSW scheme. Collusion resistance and frame-proofness of content *and* receiver tables follows from the collusion resistance and frame-proofness of the instantiated fingerprinting scheme.

Security of the Chameleon Encryption Scheme. In the technical report [22] we reduce the security of our Chameleon scheme $\mathcal{CE}$ to that of the

PRSG with which it is instantiated. To do so, we prove that no efficient algorithm can distinguish the key stream produced by $\mathcal{CE}$ from a truly random key stream.

In the remainder of this section we give the results of the security analysis, whereas further details and the proofs appear in [22]. As pointed out in Remark 1 there is a one-to-one mapping between the actual plaintext symbol space $[0, z]$ containing real numbers with finite precision and the scaled space $\{0, 1, \ldots, Z - 1\}$, which enumerates the elements of $[0, z]$ starting from 0. Further, we define the key symbol space $\mathcal{K}$ to be equal to $\{0, 1, \ldots, Z - 1\}$ and therefore $p := |\mathcal{K}| = Z$. In the sequel, by key symbols we mean the elements of $\mathcal{K}$.

Definition 1. *Let U be a uniformly distributed key symbol. Let p_k denote the probability $\mathsf{Pr}\left[X^{(1)} = x_k\right]$ of drawing key symbol $x_k \in \mathcal{K}$ in a single draw $X^{(1)}$ from master table MT. Let the* statistical quality *$SQ^{(1)}$ of MT be the statistical difference between $X^{(1)}$ and U: $SQ^{(1)} := \frac{1}{2}\sum_{k=0}^{Z-1}\left|p_k - \frac{1}{Z}\right|$. Then we call the master table* strongly converging *if $2SQ^{(1)} \leq d$ for some $d \in \mathbb{R}$ such that $d < 1$.*

Theorem 3. *Let X_k denote the k-th draw from master table MT and $X^{(s)}$ the random variable resulting from s independent uniformly distributed draws added modulo Z: $X^{(s)} := \sum_{k=1}^{s} X_k \bmod Z$. Let MT be a strongly converging master table. Then $X^{(s)}$ has a negligible statistical difference $SQ^{(s)}$ from U, where $SQ^{(s)}$ has an upper bound of $\frac{1}{2}d^s$. In addition, no probabilistic polynomial-time adversary can distinguish the key stream of $\mathcal{CE}$ from a truly random key stream.*

Implementation. We discuss implementation aspects and practical parameter choices, which allow to trade off storage size for computation cost, in [22].

5 Conclusion

In this document we gave a formal proof of the security of a new Chameleon cipher. Applied to a generic fingercasting approach, it provides confidentiality of ciphertext, traceability of content and keys as well as renewability. We achieved confidentiality through a combination of a generic broadcast encryption (BE) scheme and the new Chameleon cipher. In addition, we obtained traceability of keys and content through embedding of a receiver-specific fingerprint into the master table copies, which are given to the receivers. Finally, we achieved renewability through revocation, which is performed in the BE scheme.

References

1. Touretzky, D.S.: Gallery of CSS descramblers. Webpage, Computer Science Department of Carnegie Mellon University (2000) URL `http://www.cs.cmu.edu/~dst/DeCSS/Gallery` (November 17, 2005).
2. 4C Entity, LLC: CPPM specification—introduction and common cryptographic elements. Specification, Revision 1.0 (2003)
3. AACS Licensing Administrator: Advanced access content system (AACS): Introduction and common cryptographic elements. Specification, Revision 0.90 (2005)

4. Fiat, A., Naor, M.: Broadcast encryption. In Stinson, D.R., ed.: CRYPTO 1993. Volume 773 of LNCS, Springer (1994) 480–491
5. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In Kilian, J., ed.: CRYPTO 2001. Volume 2139 of LNCS, Springer (2001) 41–62
6. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In Yung, M., ed.: CRYPTO 2002. Volume 2442 of LNCS, Springer (2002) 47–60
7. Jho, N.S., Hwang, J.Y., Cheon, J.H., Kim, M.H., Lee, D.H., Yoo, E.S.: One-way chain based broadcast encryption schemes. In Cramer, R., ed.: EUROCRYPT 2005. Volume 3494 of LNCS, Springer (2005) 559–574
8. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In Desmedt, Y., ed.: CRYPTO 1994. Volume 839 of LNCS, Springer (1994) 257–270
9. Naor, M., Pinkas, B.: Threshold traitor tracing. In Krawczyk, H., ed.: CRYPTO 1998. Volume 1462 of LNCS, Springer (1998) 502–517
10. Kundur, D., Karthik, K.: Video fingerprinting and encryption principles for digital rights management. Proceedings of the IEEE **92**(6) (2004) 918–932
11. Anderson, R.J., Manifavas, C.: Chameleon—a new kind of stream cipher. In Biham, E., ed.: FSE 1997. Volume 1267 of LNCS, Springer (1997) 107–113
12. Briscoe, B., Fairman, I.: Nark: Receiver-based multicast non-repudiation and key management. In: ACM EC 1999, ACM Press (1999) 22–30
13. Cox, I.J., Kilian, J., Leighton, T., Shamoon, T.: Secure spread spectrum watermarking for multimedia. IEEE Trans. Image Process. **6**(12) (1997) 1673–1687
14. Kilian, J., Leighton, F.T., Matheson, L.R., Shamoon, T.G., Tarjan, R.E., Zane, F.: Resistance of digital watermarks to collusive attacks. Technical Report TR-585-98, Princeton University, Department of Computer Science (1998)
15. Anderson, R.J., Kuhn, M.: Tamper resistance—a cautionary note. In Tygar, D., ed.: USENIX Electronic Commerce 1996, USENIX (1996) 1–11
16. Maurer, U.: Conditionally-perfect secrecy and a provably-secure randomized cipher. Journal of Cryptology **5**(1) (1992) 53–66
17. Ferguson, N., Schneier, B., Wagner, D.: Security weaknesses in a randomized stream cipher. In Dawson, E., Clark, A., Boyd, C., eds.: ACISP 2000. Volume 1841 of LNCS, Springer (2000) 234–241
18. Ergün, F., Kilian, J., Kumar, R.: A note on the limits of collusion-resistant watermarks. In Stern, J., ed.: EUROCRYPT 1999. Volume 1592 of LNCS, Springer (1999) 140–149
19. Brown, I., Perkins, C., Crowcroft, J.: Watercasting: Distributed watermarking of multicast media. In Rizzo, L., Fdida, S., eds.: Networked Group Communication 1999. Volume 1736 of LNCS, Springer (1999) 286–300
20. Parviainen, R., Parnes, P.: Large scale distributed watermarking of multicast media through encryption. In: Communications and Multimedia Security (CMS 2001). Volume 192 of IFIP Conference Proceedings, Kluwer (2001) 149–158
21. Luh, W., Kundur, D.: New paradigms for effective multicasting and fingerprinting of entertainment media. IEEE Communications Magazine **43**(5) (2005) 77–84
22. Adelsbach, A., Huber, U., Sadeghi, A.R.: Fingercasting—joint fingerprinting and decryption of broadcast messages. Technical Report, Horst Görtz Institute for IT Security (2006) `http://www.prosec.rub.de/publications`.

More on Stand-Alone and Setup-Free Verifiably Committed Signatures

Huafei Zhu and Feng Bao

Institute for Infocomm Research, A-star, Singapore
{huafei, baofeng}@i2r.a-star.edu.sg

Abstract. Two notions regarding fair exchange protocols have been introduced and formalized in the literature − one is verifiably encrypted signatures; the other is verifiably committed signatures. Thus it is always interesting to explore relationship between two notions. In this paper, we first show that the existence of verifiably encrypted signatures implies the existence of the verifiably committed signatures while the existence of verifiably committed signatures does not imply the existence of verifiably encrypted signatures. As a result, the notion of verifiably committed signatures is a general extension of the notion of verifiably encrypted signatures.

The state-of-the-art verifiably committed signature that enjoys the off-line, setup-free and stand-alone properties is due to Zhu and Bao [21]. The main criticism of their paper is the use of Boudot's protocol which is pretty expensive indeed. This paper further makes contributions regarding the removal of Boudot's protocol from their construction [21]. To cope with this challenge problem, we provide a general construction of stand-alone and setup-free verifiably committed signatures from Schnorr's signature without the help of Boudot's protocol. We show that our implementation is provably secure in the random oracle model assuming that the underlying Schnorr's signature scheme is secure against adaptive chosen message attack and Paillier's encryption scheme is one-way. Since Cramer-Shoup's trapdoor hash signature is of ad hoc structure, we can embed the discrete logarithm structure where Schnorr's signature is defined into Cramer-Shoup's scheme and then apply the proved result to the verifiably committed signature of [21].

Keywords: Fair-exchange, verifiably committed signature, verifiably encrypted signature.

1 Introduction

Two notions regarding fair exchange protocols have been introduced and formalized in the literature − one is verifiably encrypted signatures; the other is verifiably committed signatures.

Verifiably encrypted signatures. The notion of verifiably encrypted signatures was introduced by Asokan et al [1, 2] in 1988. Verifiably encrypted signatures consist of the following things: a primary signer Alice wants to show

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 148–158, 2006.
© Springer-Verlag Berlin Heidelberg 2006

a verifier Bob that she has signed a message, but does not want Bob to possess her signature of that message, and Alice will give her signature to Bob only when she gets what she needs from Bob. Alice can achieve this by encrypting her signature using the public key of a trusted third party (TTP), and sending this to Bob along with a proof that she has given him a valid encryption of her signature. Bob can verify that Alice has signed the message, but cannot deduced any information about her signature. Later, in the protocol, if Alice is unable to or unwilling to reveal her signature, Bob can ask the trusted third party Charlie (arbitrator/TTP) to reveal Alice's signature.

Verifiably committed signatures. The notion of verifiably committed signatures was introduced by Park, Chong and Siegel[18] and later formalized by Dodis and Reyzin[13]. Verifiably committed signatures consist of the following things: a primary signer Alice can produce a partial signature to her verifier Bob; upon receiving what she needs from Bob, she can convert it to a full signature. If she refuses, a trusted third party Charlie (arbitrator) can do it for her upon the receipt of a partial signature and proper verification that Bob fulfilled his obligation to Alice.

1.1 A Gap Between Two Notions

We will provide the following two observations regarding two notions above. We show that

Claim 1. the existence of verifiably encrypted signatures implies the existence of the verifiably committed signatures;

Claim 2. the existence of verifiably committed signatures does not imply the existence of verifiably encrypted signatures.

To see the first claim, we consider the following verifiably encrypted signatures below[8]:

Step 1: Alice sends Bob her verifiable encryption of a signature, attaching a condition that describes the signature that Bob is supposed to give Alice;

Step 2: Bob receives and verifies the verifiable encryption of the signature from Alice. If this succeeds (e.g, the verification algorithm outputs 1), Bob sends Alice his signature (an ordinary signature on the message); otherwise, Bob quits;

Step 3: Alice receives and verifies the ordinary signature from Bob. If this succeeds, Alice sends Bob her signature, and quit;

Step 4: Bob receives and verifies the signature from Alice. If this succeeds, Bob outputs the signature sent by Alice and quit; otherwise, Bob takes the received message in Step 1 and a proof that he fulfilled his obligation to Alice in Step 2 to the trust third party for obtaining a valid full signature of Alice on the message.

We now simply define Alice's verifiably encrypted signature as her partial signature and her signature sent to Bob in the Step 3 as her full signature (in terms of verifiably committed signatures). Thus, the existence of verifiably encrypted signatures implies the existence of the verifiably committed signatures.

To see the second claim, we consider the illustrative counterexample below: suppose a primary signer has public/secret key (pk_1, sk_1) for the first signature scheme, and at the same time the prime signer and its TTP share another public/secret key (pk_2, sk_2) of the second signature scheme. By $PK = (pk_1, pk_2)$ we denote the public key of the entire signature scheme, and by $SK = (sk_1, sk_2)$, we denote the corresponding secret keys. Now given a message m, the primary signer produces its partial signature σ_1 on the message m. A full signature of the message m is defined as $\sigma = (\sigma_1, \sigma_2)$, where σ_2 is the signature of m correspondent the public/secret key pair (pk_2, sk_2). It is easy to verify that the resulting signature is a verifiably committed signature scheme. Obviously, there is no semantically secure public key encryption scheme involved in the above counterexample (thus the verifiably committed signature scheme described above is not a verifiably encrypted signature scheme).

Summary. In summary, we have the following claim:*the notion of verifiably committed signatures is a general extension of the notion of verifiably encrypted signatures.*

1.2 More on Verifiably Committed Signatures

Informally, a verifiably committed signature consists of three participants: a primary signer, a verifier and a trusted third party (TTP). TTP can be on-line or off-line (the advantage of the off-line protocol to the on-line protocol is obvious). We provide further observations regarding off-line verifiably committed signatures below:

Off-line verifiably committed signatures can be classified into two categories: with or without initial-key-setup procedures. An off-line verifiably committed signature is called setup-free if no initial-key-setup procedure between a primary signer and its TTP is involved except for one requirement that the primary signer can obtain and verify TTP's certificate and vice versa. An off-line verifiably committed signature is called setup-driven if an initial-key-setup protocol between a primary signer and its TTP must be involved such that at the end of the key setup protocol, the primary signer and its TTP share prior auxiliary information. This shared auxiliary information enables TTP to convert any valid partial signature into the corresponding full signature if a confliction occurs between the primary signer and its verifier.

Another interesting feature of verifiably committed signatures is stand-alone property. The stand-alone property means that the resulting signature (from an off-line fair exchange) is identical as one of an ordinary signature scheme. From the point of views of both practical application and theoretical research, any verifiably committed signature with off-line, step-free and stand-alone properties is certainly welcome.

1.3 Previous Works and Problem Statement

The research of fair exchange protocols has a rich history due to its fundamental importance (see [5, 8, 10, 14, 15, 16] for general references). Several works have

been done within the setup-driven model (e.g., [7], [20], [18] and [13]). We stress that the definition of verifiably encrypted signatures in [1, 2] implies the off-line, setup-free and stand-alone properties inherently but it does not imply that a verifiably committed signature satisfies setup-free and stand-alone properties inherently (e.g., the recent work of Boneh, Gentry, Lynn and Shacham [6] is a two-signature based verifiably committed signature scheme thus the stand-alone property of the BGLS protocol is completely NOT satisfied ($e(\omega, g_2) = e(h, \upsilon)$ $e(\mu, \upsilon')$)).

The state-of-the-art verifiably committed signature that enjoys the off-line, setup-free and stand-alone properties is due to Zhu and Bao [21]. Their scheme is based on the Cramer-Shoup's signature scheme. The main criticism of Zhu and Bao's paper is the use of Boudot's protocol which is pretty expensive indeed. Based on this observation, we further presented a challenge problem below:

Question (suggested by Dodis): Is it possible to remove Boudot's protocol from Zhu and Bao's construction [21]?

1.4 This Work

In this paper, we provide a positive solution to the challenge problem suggested by Dodis if stand-alone, setup-free verifiably committed signatures presented in [21] are constructed form Cramer-Shoup's trapdoor hash signatures. The general idea behind our solution is that − we first provide a general construction of stand-alone and setup-free verifiably committed signatures from Schnorr's signature without the help of Boudot's protocol, and show that our implementation is provably secure in the random oracle model assuming that the underlying Schnorr's signature scheme is secure against adaptive chosen message attack and Paillier's encryption scheme is one-way. Since Cramer-Shoup's trapdoor hash signature is of ad hoc structure, we can embed the discrete logarithm structure where Schnorr's signature is defined into Cramer-Shoup's scheme and then apply the proved result to the verifiably committed signature of [21].

Limitation of our approach: we stress that the results presented in [21]cover two cases:

Case 1: stand-alone, setup-free verifiably committed signatures constructed from Cramer-Shoup's trapdoor hash signatures;
Case 2: stand-alone, setup-free verifiably committed signatures constructed from Zhu's signatures [20];

We remark that our approach presented in this paper cannot be adopted to deal with Case 2 since a primary signer must prove that a salt used to generate a valid signature lies in the correct interval (e.g., $t \in \{0, 1\}^l$). How to remove Boudot's protocol from Zhu-Bao's construction if underlying signature scheme is Zhu's signature scheme is still unknown. This leaves an open problem to the research community.

1.5 Road Map

The rest of the paper is organized as follows: In section 2, syntax and security definition of off-line, stand-alone and setup-free verifiably committed signatures are briefly introduced. In Section 3, the building blocks that will be used for the construction of the Dlog-based scheme are sketched; In section 4, an efficient implementation of verifiably committed signatures (from Schnorr's signature schemes) as well as the proof of the security are proposed. We embed the discrete logarithm structure where Schnorr's signatures are defined into Cramer-Shoup's scheme and then apply the proved result to the verifiably committed signature of [21] is presented in Section 5. We conclude our works in Section 6.

2 Syntax and Security Definition

2.1 Syntax

This paper presents a verifiably committed signature which has the following properties: first it is setup-free, which means that no interaction is required between a signer and its TTP except for an exchange of certificates; second it is stand-alone, which means that the resulting signature (from an off-line fair exchange) is identical as one of an ordinary signature scheme. The formal definition of stand-alone and setup-free verifiably committed signatures has been formalized by Zhu and Bao and we refer to the reader [21] for more details.

2.2 The Definition of Security

Recall that the concept of verifiably committed signatures is formalized the same thing as off-line fair-exchange protocols. Thus the security definition of a verifiably committed signature should consist of three aspects: security against a primary signer Alice, security against a verifier Bob, and security against a arbitrator/TTP Charlie. The security definitions of verifiably committed signatures follows Dodis and Reyzin's and we thus refer to the reader [13] for more details).

3 Building Blocks

3.1 Schnorr Signature Scheme

The Schnorr signature scheme [19] sketched below, was first proposed as an application of the Fiat-Shamir transformation. It can be instantiated on any group G of prime order in which the discrete logarithm problem is believed to be hard. Schnorr signature scheme has been proven secure under the standard notion of existential unforgeability against an adaptive chosen-message attack in the random oracle model.

- The key generation algorithm KG: On input 1^k, it produces two large primes p, q such that $q|(p-1)$, and an element $g \in Z_p^*$ of order q. Then it picks a random element $x \in Z_q^*$, and sets $y = g^x \bmod p$. The public key is $\{p, q, g, y, H\}$, where H is a cryptographic hash function, mapping arbitrary strings to the elements of Z_q, while the correspondent secret key is x.
- The signing algorithm Sig: To sign a message m, the signer picks a random $t \in Z_q^*$, and sets $r = g^t \bmod p$. It then computes $e = H(m, r)$, $s = t + xe \bmod q$, and outputs the signature $\sigma = (r, s)$.
- The verification algorithm Ver: To check whether a given (r, s) is indeed a signature of some message m, it suffice to know the correspondent public key $\{p, q, g, y, H\}$ and verify that $g^s = ry^e \bmod p$, where $e = H(m, r)$.

3.2 Paillier's Cryptographic System

Paillier investigated a novel computational problem, called Composite Residuosity Class Problem, and its applications to public key cryptography in [17]. Our construction will heavily rely on this probabilistic encryption scheme which is sketched below.

The public key is a k_1-bit RSA modulus $N = PQ$, where P, Q are two large safe primes. The plain-text space is Z_N and the cipher-text space is $Z_{N^2}^*$. To encrypt $\alpha \in Z_N$, one chooses $r \in Z_N^*$ uniformly at random and computes the cipher-text as $E_{PK}(a, r) = g^a r^N \bmod N^2$, where $g = (1 + N)$ has order N in $Z_{N^2}^*$. The private key is (P, Q).

It is straightforward to verify that given $c = (1+N)^a r^N \bmod N^2$, and trapdoor information (P, Q), one can first computes $c_1 := c \bmod N$, and then compute r from the equation $r = c_1^{N^{-1} \bmod \phi(N)} \bmod N$; Finally, one can compute a from the equation $cr^{-N} \bmod N^2 = 1 + aN$.

The encryption function is homomorphic, i.e., $E_{PK}(a_1, r_1) \times E_{PK}(a_2, r_2) \bmod N^2 = E_{PK}(a_1 + a_2 \bmod N, r_1 \times r_2 \bmod N)$.

3.3 Proof of Knowledge of Encryptions

We make use of the following protocols for proof of knowledge of encryptions. An efficient implementation has been proposed by Damgård and Jurik[11] can be tailored for our purpose:

- Let λ be maximum bit length of x. Suppose $u = g^x$ is defined over G, where p, q are two large primes such that $q|(p-1)$, and an element $g \in Z_p^*$ of order q. Given a cipher-text $c = E_{PK}(x)$ which is computed from Paillier's encryption scheme, the prover should provide a proof that u and c hide the same value x.
- The prover chooses at random $\omega \in \{0, 1\}^{\lambda+2l}$, where l is a security parameter. The prover sends $a = E_{PK}(\omega)$ and $b = g^\omega$ to the verifier. Here we assume that the security parameter k_1 of Paillier's system is larger than $(\lambda + 2l)$
- The verifier chooses a l-bit challenge f;

– The prover opens the encryptions $ac^f \bmod N^2$ and $bu^f \bmod p$, to reveal in both cases the number $z = \omega + xf$ defined over the integer domain. The verifier checks the opening were correct.

We stress that the proof of knowledge of encryptions $u = g^x$ and $c = E_{PK}(x)$, does not ensure that u and c hide the same value. In fact, for any value $a = x \bmod q$ which lies in the interval $(\lambda + 2l)$ may satisfy the equation since q is a publicly known value if follows that once given a one can easily compute the exact value x that lies in the interval $[0, q-1]$. We also stress that one can not compute $b = x \bmod N$ at the same time since we assume that $(\lambda + 2l) \leq N$. In the following argument, we extract of knowledge of encryptions in the sense of above.

4 Verifiably Committed Signatures with Dlog Structures

4.1 The Implementation

In this section, we describe our implementation of stand-alone and setup-free verifiably committed signature scheme from Schnorr's signatures.

– Primary signer key generation algorithm: on input 1^{k_1}, a probabilistic polynomial time primary signer Alice generates two large safe primes p and q such that $q|(p-1)$. Let $G \subset Z_p^*$ with order q and g be a generator of G, i.e., $< g > = G$. Alice also chooses a random element $x \in [1, q]$ to compute her public key $y = g^x \bmod p$. The public key of a primary signer is (p, q, g, y, H), where H: $\{0,1\}^* \rightarrow [1, q]$ is a collision-free hash function. The private key is x. Alice also proves to her CA that all values are correctly generated and then obtains her certificate $cert_A$ from her CA;
– Cosigner key generation algorithm: on input 1^{k_2}, a probabilistic polynomial time cosigner Charlie generates a k_2-bit RSA modulus $n = p_c q_c$, where p_c, q_c are two large safe primes. The public key is $((1+n), n)$, and the private key is (p_c, q_c). Charlie also proves to his CA that all values are correctly generated and then obtains his certificate $cert_C$ from his CA;
– Fully signing algorithm Sig and its correspondent verification algorithm Ver: To sign a message m, the signer picks a random $s \in Z_q^*$, and sets $u = g^s \bmod p$. It then computes $e = H(m, u)$, $t = s + xe \bmod q$, and outputs the signature $\sigma = (u, t)$. The verification algorithm Ver: To check whether a given (u, t) is indeed a signature of some message m, it suffice to know the correspondent public key $\{p, q, g, y, H\}$ and verify that $g^t = uy^e \bmod p$, where $e = H(m, u)$.
– Partially signing algorithm $PSig$: The partial signature σ' is (u, v, w, pr). That is, to generated a partial signature Alice first chooses a random string $s \in z_q$ and then computes $t = s + xe$, $v = g^t$ and $w = E_{pk_C}(t) = (1+n)^t r_t^n \bmod n^2$ such that $v = uy^e \bmod p$ and a proof pr that both v and w hide the same value t by means of the proof of knowledge of encryption (see section 3 for more details). We stress that the proof can be interactive and

non-interactive (a transform from an interactive proof to the correspondent non-interactive can be trivially implemented by the standard Fiat-Shamir's technique).

- The correspondent partial signature verification algorithm $PVer$: Given a putative signature $\sigma'= (u, v, w, pr)$, the verifier Bob checks the validity of the equation $v = uy^e$, and if the equation is valid then Bob further checks the validity of the proof (a proof to show that both v and w hide the same value t). If both both are valid then Bob accepts (the partial signature is valid), otherwise, it rejects.
- Resolution algorithm Res: Given $\sigma'=(u, v, w, pr)$ and a proof that Bob fulfilled his obligation to the primary signer. If the verification is passed, then Charlie outputs a valid full signature of (u, t) using his decryption key, otherwise, it rejects.

4.2 The Proof of Security

In this section, we will prove that our construction is secure in the sense of section 2. That is,

Lemma 1. *Our construction is provably secure against a primary signer Alice.*

Proof: Suppose Alice is able to provide a valid partial signature $\sigma' = (u, v, w, pr)$ correspondent to a message m. Since σ' is valid from the point views of both the verifier and the cosigner, by rewinding Alice, both verifier and cosigner can extract t such that $v=g^t$, $w=E_{pk_C}(t)$ and $g^t = uy^e \bmod p$, where $e = H(m, u)$ (without loss of generality, we here simply assume that v and w hide the same value t, see Section3.3 for more details). It follows that the cosigner can always transform any valid partial signature scheme (u, v, w, pr) into the correspondent valid signature $\sigma=(u, t)$ of the same message m.

Lemma 2. *Our construction is provably secure against a verifier Bob if the underlying Paillier's encryption scheme is one-way.*

Proof: By $\mathcal{B}$, we denote an attacker that attacks our verifiably committed signature scheme. By $\mathcal{B}'$, we denote an inverter of a random cipher-text generated by the underlying encryption scheme. $\mathcal{B}'$ simulates the environment of $\mathcal{B}$ as follows. First it runs Alice to generate primary signer's public key and secret key same as that in the real protocol and outputs (PK, SK); Then it runs Charlie to generate a cosigner's public key and secret key such that ASK is not an output to $\mathcal{B}'$. That is, PK and APK are given to $\mathcal{B}$, and $\mathcal{B}'$ obtains (PK, SK) and APK. Let (m^*, σ^*) be a successful forgery of the attacker $\mathcal{B}$, without loss of generality, we assume that $\mathcal{B}$ obtains (m^*, σ'^*) from the partial signing oracle $PSig$ since the underlying Schnorr's signature scheme is secure against adaptive chosen-message attack. Let q_{Psig} be the total number of P queries made by $\mathcal{B}$, and let ξ be a random number chosen from $\{1, q_{Psig}\}$ by $\mathcal{B}'$. We now simulate $PSig$ oracle queries as follows: 1) if $i \neq \xi$, then $\mathcal{B}'$ runs the partial signing oracle as the real partial signature scheme; 2) if $i = \xi$, for a given target cipher-text

w, $\mathcal{B}'$ chooses random strings $s \in Z_q$ then computes $u = g^s$ and $v = g^t$, where $t = s + xe$, $e = H(m, u)$.

The rest work is simulate that v and w hide the same value. To do so, the simulator chooses a random string $\bar{t}$, and computes $\bar{v} = g^{\bar{t}}$. The simulator then computes $\bar{w}$ from the equation $E_{pk_C}(z_t) = \bar{w}\ w^f$, where f is a random string distributed over domain G, and $z_t = \bar{t} + tf$ which is computed from the integer domain; 3) finally $\mathcal{B}'$ assigns f be a hash value of the random oracle; $\mathcal{B}'$ simulates R oracle queries as follows: 1) for (m_j, σ'_j) that is in the partial signature query list, if $j \neq \xi$, then R outputs t_i, otherwise it aborts; 2) for (m_j, σ'_j) that is NOT in the partial signature query list, then R outputs $\perp$; Notice that the probability that the simulator does not abort is at $1 - 1/q_{Psig}$, thus when the adversary outputs a successfully forgery (m^*, σ^*) whose partial signature is the list of P oracle query, then the probability that $\mathcal{B}'$ invert w with probability at least ϵ/q_{Psig}, where $\mathcal{B}$ forgers successfully a forgery with probability ϵ.

Lemma 3. *Our construction is provably secure against a arbitrator/cosigner Charlie assuming that the underlying Schnorr scheme is secure against adaptive chosen-message attack.*

Proof: Suppose Charlie is able to forgery partial signature σ' with non-negligible probability, then by rewinding Charlie, an adversary can extract t from pr, and thus, Charlie is able to forge a valid signature of the Schnorr's signature scheme with non-negligible probability. This is a contradiction.

In summary, we have proved the main result below:

Theorem 1. *The stand-alone and setup-free verifiably committed signature scheme constructed above is provably secure assuming that the underlying Schnorr's signature scheme is secure against adaptive chosen-message attack, and Paillier's encryption is one-way.*

5 Embedding the Dlog Structure to Cramer-Shoup's Trapdoor Hash Signatures

5.1 Cramer-Shoup's Trapdoor Hash Signatures

Cramer-Shoup's signature scheme (CS) is one of the fastest and few known signatures without involving random oracle and it would be faster than two non-random oracle based signatures. To embed our proved result above into the CS scheme, we first sketch this notable scheme below [9]:

- Key generation algorithm: Let p, q be two large safe primes such that $p-1 = 2p'$ and $q-1 = 2q'$, where p', q' are two l'-bit primes. Let $n = pq$ and QR_n be the quadratic residue of Z_n^*. Let x, h be two generators of QR_n. Also chosen are a group G of order s, where s is $(l+1)$-bit prime, and two random generators g_1, g_2 of G. The public key is (n, h, x, g_1, g_2, H) along with an appropriate description of G including s. The private key is (p, q).

– Signature algorithm: To sign a message m, a $(l+1)$-bit prime e and a string $t \in Z_s$ is chosen at random. The equation $y^e = xh^{H(g_1^t g_2^{H(m)})} \mathrm{mod} n$ is solved for y. The corresponding signature of the message m is (e, t, y), where the variable $g_1^t g_2^{H(m)}$ is a virtual commitment.
– Verification algorithm: given a putative triple (e, t, y), the verifier first checks that e is an odd $(l+1)$-bit number. Second it checks the validation that $x = y^e h^{-H(g_1^t g_2^{H(m)})} \mathrm{mod} n$. If the equation is valid, then the verifier accepts, otherwise, it rejects.

5.2 An Ad-Hoc Solution

We stress that the choice of a group G described in the CS scheme is independent of the choice of RSA moduli. Thus, we simply define G used in the CS scheme is the same group where Schnorr's signature scheme is defined. Based on this observation provide an ad-hoc solution below:

Full signature: the full signature is defined as ordinary signature of the CS scheme;

Partial signature: the partial signature is defined as follows: on input a message m, $(l+1)$-bit prime e and a string $t \in G$ which is chosen at random. The equation $y^e = xh^{H(g_1^t g_2^{H(m)})} \mod n$ is solved for y. Then Alice hides t by computing $u{=}g_1^t$ and $c{=}E_{APK}(t)$ together with a proof pr that u and c hide the same value x by means of Damgd̊ and Jurik's protocol specified in Section 3. The partial signature is defined by $\sigma'{=}(e, y, u, c, pr)$.

Resolution algorithm: the resolution algorithm is the same as that presented in protocol constructed from Schnorr's signature (see Section 4 for more details).

As an immediately application of Theorem 1, we have the following statement:

Corollary 1. *It is possible to remove Boudot's protocol from Zhu and Bao's construction if their verifiably committed signatures are constructed from Cramer-Shoup's trapdoor hash signatures.*

6 Conclusion

We have presented an efficient implementation of off-line, stand-alone and setup-free verifiably committed signature scheme from Schnorr signatures. We then embed the discrete logarithm structure where Schnorr's signature is defined into Cramer-Shoup's scheme. Finally, we show that there is a method to remove Boudot's protocol from Zhu and Bao's construction [21] if it is constructed from Cramer-Shoup's trapdoor hash signature.

Acknowledgment. The first author thanks Professor Yevgeniy Dodis for his contribution of the motivation problem and his continuous encouragement.

References

1. N.Asokan, M.Schunter, M.Waidner: Optimistic Protocols for Fair Exchange. ACM Conference on Computer and Communications Security 1997: 7 - 17.
2. N.Asokan, V.Shoup, M.Waidner: Optimistic Fair Exchange of Digital Signatures (Extended Abstract). EUROCRYPT 1998: 591 - 606.
3. N.Asokan, V.Shoup, M.Waidner Optimistic fair exchange of digital signatures, IEEE Journal on Selected Areas in Communications 18(4):593-610, 2000
4. F.Boudot: Efficient Proofs that a Committed Number Lies in an Interval. Proc. of EUROCRYPT 2000: 431 - 444, Springer Verlag.
5. M.Ben-Or, O.Goldreich, S.Micali and R.L.Rivest: A Fair Protocol for Signing Contracts (Extended Abstract). ICALP 1985: 43 - 52.
6. D.Boneh, C.Gentry, B.Lynn and H.Shacham: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. EUROCRYPT 2003: 416 - 432
7. C. Boyd, E. Foo: Off-Line Fair Payment Protocols Using Convertible Signatures. ASIACRYPT 1998: 271 - 285
8. J.Camenisch, V.Shoup: Practical Verifiable Encryption and Decryption of Discrete Logarithms. CRYPTO 2003: 126 - 144
9. R. Cramer and V. Shoup. Signature scheme based on the Strong RAS assumption. 6th ACM Conference on Computer and Communication Security, Singapore, ACM Press, November 1999.
10. I.Damgård: Practical and Provably Secure Release of a Secret and Exchange of Signatures. EUROCRYPT 1993: 200 - 217.
11. I.Damgård, M.Jurik: Client/Server Tradeoffs for Online Elections. Proc. of Public Key Cryptography 2002: 125 - 140. Springer Verlag.
12. I.Damgård, E.Fujisaki: A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. Proc. of ASIACRYPT 2002: 125 - 142, Springer Verlag.
13. Y.Dodis and L.Reyzin. Breaking and Repairing Optimistic Fair Exchange from PODC 2003, ACM Workshop on Digital Rights Management (DRM), October 2003.
14. J.Garay, M.Jakobsson and P.MacKenzie: Abuse-Free Optimistic Contract Signing. CRYPTO 1999: 449 - 466
15. W.Mao: Verifiable Escrowed Signature. ACISP 1997: 240 - 248
16. S. Micali: Simple and fast optimistic protocols for fair electronic exchange. PODC 2003: 12 - 19.
17. P.Paillier: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Proc. of EUROCRYPT 1999: 223 - 238, Springer Verlag.
18. J.Park, P.Chong and H.Siegel: Constructing fair-exchange protocols for E-commerce via distributed computation of RSA signatures. PODC 2003: 172 - 181
19. C.Schnorr. Efficient identification and signature for smart card. Cryptology-Crypto'89, 235-251.
20. H.Zhu: Constructing Committed Signatures from Strong-RSA Assumption in the Standard Complexity Model. Public Key Cryptography 2004: 101 - 114.
21. H.Zhu, F.Bao: Stand-Alone and Setup-Free Verifiably Committed Signatures. CT-RSA 2006: 159-173

API Monitoring System for Defeating Worms and Exploits in MS-Windows System

Hung-Min Sun, Yue-Hsun Lin, and Ming-Fung Wu

Department of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan 30013
hmsun@cs.nthu.edu.tw, {tenma, coolflame}@is.cs.nthu.edu.tw

Abstract. Worms and Exploits attacks are currently the most prevalent security problems; they are responsible for over half of the CERT advisories issued in the last three years. To initiate an infection or intrusion, both of them inject a small piece of malicious code (ShellCode) into software through buffer or heap overflow vulnerabilities. Unlike Unix-like operating systems, ShellCodes for Microsoft Windows system need more complex steps to acquire Win32 API calls from DLL file (Dynamic Load Library) in Microsoft Windows. In this paper, we proposed an effective API monitoring system to get rid of worms and exploits attacks for the Microsoft Windows without hardware support. We address the problem by noticing that ShellCodes need the extra complex steps in accessing Win32 API calls. Through the API monitoring system we purposed, we can successfully stop the attacks made by worms and exploits. Moreover, the efficiency of Win32 API Calls hooking and monitoring system can be improved. Incapability to disassemble and analysis the protected software processes are overcome as well.

Keywords: Worm Protection, System Security, API Hooking.

1 Introduction

From the last decade, computers have been threaten by worms and exploits attacks. General speaking, worms and exploits are some “harmful code” implanted in victim computer to hack secure data and widespread the “harmful code” to another machine by any other means [11, 12]. They bring large damages to the world from finances to militaries. It has been wars between security-expects and hackers in defending and creating new worms and exploits.

Microsoft Windows systems are the most popular operating system series nowadays. Perhaps, owing to its closure of source code, hackers have some advantages over security-expects in developing new worms and exploits while the latter has not enough time and knowledge to react. For its popularity and hardness of making up patches, attacks of worms and exploits incur significant consequences which raise the interest of researches. Unless explicitly declare, we will focus on Microsoft Windows system in the rest of the paper.

A passive movement against worms and exploits attacks is waiting for the new release of the code and writing new patches. It not only involves great humanity

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 159 – 170, 2006.
© Springer-Verlag Berlin Heidelberg 2006

resources, but also induces damages until the time when new patches are released. Otherwise, an active movement is to develop a prevention system as a safeguard which monitors the behaviors of all running processes. This safeguard should not be codes dependent, i.e., it should prohibit all malicious behaviors of processes like formatting hard disk, or more specifically, calling some APIs. In fact, the active mechanism has better performance to resist malicious code.

In this paper, we develop an active mechanism, a monitoring system in windows environment to protect software process from injecting malicious codes. The main advantage of our method is that it does not stick to any type of code injection and can be applied to Microsoft Windows NT series Operating Systems such as Windows 2000 and Windows XP, since our process can be applied to the Win32 Portable Executable File Format (PE32) [13].

The main idea of this paper is that all software including malicious codes, worms, exploits and Trojan horses, need to use the Win32 API Calls [4] to interact with the Windows operating system. If we can monitor any irregular use of these API Calls, the malicious code can then be detected and stopped.

The rest of the paper will be organized as follows. In section 2, we will go through some characteristic of ShellCodes and its variations. In section 3, we will present our monitor system. Its performance will be evaluated at section 4. In section 5, the comparisons among different types of monitoring system will be given. In section 6, we will end up with a conclusion.

2 Primitives

In this section, we will first introduce some basic characteristics of ShellCodes. Then we will classify different kinds of ShellCodes.

2.1 Basic Aspect of Shellcodes

Both worms and exploits need a hook to download them from the network to the vulnerable machine, and then execute or work to control the target. After that, the attacker may be able to destroy the system or to raise DDoS attack [5, 7]. These hooks are called ShellCodes.

According to our studies, a successful injection of malicious code will require three conditions [3, 19, 22-24]:

- Overflow Vulnerabilities: Hackers may find some security holes in software that are vulnerable to buffer or heap overflow attacks. Therefore, Hackers can overwrite the return address of that function and inject the malicious code via these holes.
- Return Address Overwrite: To run the injected code correctly, ShellCodes must overwrite the return address of the vulnerable function exactly where the malicious code starts and overwrite the address of EIP.
- Independent execution**:** Usually, ShellCodes must call Win32 APIs when they implant the target system. However, ShellCodes do not know where the APIs are since they are not loaded by the loader but injected into software process at runtime. ShellCodes have to get APIs' addresses without the help of loader.

The ShellCodes would be success when all three conditions hold.

2.2 Classification of Shellcodes

ShellCodes can be grouped into three types by how they find windows native APIs' addresses [21] to control the system. Each type will be described in the following subsections:

1) Manual Input Addresses of API Calls

The direct way for hackers to get the addresses of API Calls is searching their own computers to get APIs' addresses and assign these APIs' addresses to an array in ShellCodes [23]. The program code is showed in Fig. 1.

In fact, hackers only need to input two API addresses manually into ShellCodes in order to use them to get other APIs in Windows. These two API addresses are "LoadLibraryA" and "GetProcAddress" [3, 19, 22-24]. The main drawback of this kind of ShellCodes is that they cannot run correctly on different version of operating systems, such as different language versions of Microsoft Window XP. This will decrease the probability of successful attack.

```
loadlibfun=API_ARRAY[0]  //address of LoadLibraryA
__asm{

lea eax,user32
push eax
call dword ptr loadlibfun
//LoadLibrary("user32");
}
```

Fig. 1. The program code to get the address of user32.dll in the local machine

2) Scan Memory to Find API Addresses

To improve the portability of ShellCodes, hackers try to find the API addresses by scanning the memory space. These Win32 APIs are placed in some DLLs (Dynamic Link Libraries) that are loaded to memory with 64k boundary[1]. DLL files are also PE32 files [13]. When the DLL is loaded into memory, its memory segment has a pair of tags "MZ" and "PE" inside. Therefore, the ShellCodes can scan memory address above 0x70000000h[2] to find the "MZ" and "PE" tag in order to locate the start address of DLLs [23].

After the ShellCode has retrieved the starting address of the DLL, it can locate APIs' addresses in the ExportTable in where to obtain the address of API function using similar technique as type 1. However, this technique has its limitation. While ShellCodes scanning memory to find API addresses, GPF (General Protect Fault) might probably occur. The ShellCodes will then be terminated.

3) Get ImageBase of DLLs from TEB

The last kind of ShellCodes can get the ImageBase of kernel32.dll which is the base memory address of kernel32.dll from the Thread Environment Block (TEB) directly

[1] The DLL address must align with 0x00010000h.

[2] Base address of kernel32.dll is always placed above 0x70000000h in Windows NT series. For example, the base address of kernel32.dll in Windows XP is 0x77E60000h.

without scanning memory or inputting them manually. Most of recent worms and exploits are in this form. We can further subcategorize two ways of getting the kernel32.dll from the TEB:

a) Get ImageBase of DLLs from PEB
The kernel32.dll base address can be determined through the Process Environment Block (PEB). Every process has a PEB to record environment variables. The PEB of every process can always be found at fs: 0x30 within the process. The address of the kernel32.dll can be obtained from the PEB structure through the following steps:

1st: Find the memory address of TEB from the fs pointer (fs: 0).
2nd: Find the address of PEB structure at fs: 0x30h (0x30h offset from fs: 0).
3rd: Find the address of PEB_LDR_DATA structure at offset 0x0c from PEB structure (PEB+0x0c).
4th: Find the start address of the DLLs list at offset 0x1c from PEB_LDR_DATA structure (PEB_LDR_DATA+0x1c).
5th: Search DLL module list and find the address of kernel32.dll.

b) Get ImageBase of DLLs from SEH (Structured Exception Handling)
The other way to obtain the base address of kernel32.dll depends on the SEH structure. SEH is a mechanism for handling both hardware and software exceptions. Since the default Unhandled Exception Handler is set to use a function that exists inside kernel32.dll, the address of the kernel32.dll can be obtained in the following steps:

1st: Find the memory address of TEB from the fs: 0 as before.
2nd: Find the address of *pvExcept* from TEB for which will point to a series of exception handler (SEH).
3rd: Iteratively search the SEH until the end. The end of SEH is the unhandled exception handler.
4th: The address of unhandled exception handler can be used as a starting point for walking down in increments of 64KB.

As we mentioned before, the addresses of DLLs will only align on 64KB boundaries. At each 64KB boundary, we can check whether it is a DLL file. Once a match is found it is safe to assume that the base address for kernel32.dll has been found.

3 The Proposed Approach

Basically, our approach can be divided into four parts: (A) PE loader method: It creates our own loader for monitoring and memory allocation. (B) Prevent ShellCodes to get the address of Kernel32.dll from SEH and PEB structure: It prevents the ShellCodes from getting the right API addresses from SEH and PEB in TEB. (C) API Hooking and Monitoring: We provide an interface between API calls and running process to prevent ShellCodes to get the real address of API calls. (D) Rebasing method: It uses the relocation method to rebase (relocate) the address of Kernel32.dll.

These methods not only allow program that already compiled to binary be correctly executed, but also the software itself is encrypted for software protection can also be performed.

3.1 PE Loader Method

To protect software processes from being attacked by worms and exploits, we have to change the location where the processes would be loaded into the memory and vary the stack memory. In general, most of ShellCodes are about 300~1000 bytes. If our variation is bigger than 1000 bytes, it would be difficult for worms and exploits to hit the right position of ShellCodes in stack memory.

1) Change the Location of Software Processes (*CLSP*)

Unlike DLL Files, executable files (EXE) do not have relocation tables for us to rebase their ImageBase address. Therefore, we decide to use our own PE loader [13] to adjust where the processes would be placed.

Our loader starts by checking PE header and setting the StartupInfo of the executable file that we wanted to load. After checking the validity of all the headers in the executable file, we mapped and loaded the process file into the memory location we allocated manually. Finally, we created a remote thread from where we allocated and started from there.

The only problem is how to replace the original PE Loader from Software processes that were locked by Windows. By using the Win32 API call MoveEx(), we moved the running processes to other position and replaced our loader to the original place. Next time when these processes started by Windows, it will run our loader first and change the location where processes run.

2) Variations of Stack Memory Address (*VSMA*)

The stack memory addresses in windows system are not as unstable as we consider. If the order of creating processes or startup are the same, their stack address would most likely be at the same place too. This is the reason that worms and exploits can inject their ShellCodes into stack memory and run them correctly. Here are some features that would affect the stack address of each process: (1) Memory address of runtime processes, (2) Stack reserved and commit from software processes' PE header and (3) Stack size commit from each thread.

We have changed processes' memory address in the previous section. By modifying two fields, "SizeOfStackReserve" and "SizeOfStackCommit", in the PE header of software processes, we successfully influence the variations of stack memory address. After modifying the PE header, we also hook Win32 API CreateThread() and CreateRemoteThread() to change the stack size of each thread when it create. In fact, every program which uses dynamic memory allocation can be loaded and executed by our PE loader.

3.2 Prevent ShellCodes to Get the Address of Kernel32.dll from SEH and PEB Structure

After the static analysis (disassemble) of ShellCodes used by recent worms and exploits, we can find that these ShellCodes executed independently by calling API function calls. The first step is to get the base address of kernel32.dll. In the following

sections, we will illustrate two methods to prevent the address of kernel32.dll from being obtained by ShellCodes.

1) Modify SEH Handler

As mentioned in section 2.3.2, ShellCodes finds the memory address of kernel32.dll through the Unhandled Exception Handler. After getting the memory address of the unhandled Exception Handler, ShellCodes would know where the kernel32.dll's memory address is around. They can then scan the memory with 64k boundary [23].

In order to stop the ShellCodes, we change the pointer of the Unhandled Exception Handler to our own Unhandled Exception Handler, whose original memory is not in kernel32.dll's memory space. And we also keep the pNextmember as 0xffffffff, this will make ShellCodes only find the address of our own Unhandled Exception Handler and get its address then scan on wrong memory space.

2) Replace Kernel32.dll from DLL Record List

The other way for ShellCodes to get the address of kernel32.dll is through the PEB structure. PEB is the structure for each process to store some information about itself. The PEB can always be found at fs: 0x30 within the process which holds information about the process heaps, binary image information, and, most importantly, three linked lists regarding loaded modules that have been mapped into process space. The linked lists themselves differ in purposes from showing the order in which the modules were loaded to the order in which the modules were initialized. Most interesting is that the order of kernel32.dll is always the second module to be initialized. This explains why ShellCodes can get kernel32.dll‘s memory address easily by walking from the list to the second entry.

The module list in PEB looks more complex and significant than the record of exception handler list. Before we replace the pointer that points to kernel32.dll in the memory space, we must have a copy of original kernel32.dll. The following problems were encountered when we started to duplicate kernel32.dll:

1. If we use LoadLibrary()to load our own copy of kernel32.dll, the system would have errors, showing that functions can not have the same names.
2. If we use CreateFileMapping()to create a mapping of our own kernel32.dll in the memory, it is slightly different from using LoadLibrary() to load it into the memory.
3. File that was mapped into the memory requires relocation and realignment.

After all, we decided to dump kernel32.dll from its memory space and map the file that we have dumped into memory. This would solve the first and the second problems that we encountered above, and it would not require realignment. Subsequently, we would rebase and relocate our mapping of kernel32.dll to its new base address. After the duplication of kernel32.dll is created, we start to trace PEB just like what ShellCodes do to find where the module list is. Here are the steps of how to find and trace module list in PEB:

1st: Find the pointer to PEB header at fs: 0x30.
2nd: Find the pointer to PEB_LDR_DATA at offset 0x0c from PEB header.
3rd: Find the pointer to the module list at offset 0x1c from PEB_LDR_DATA.
4th: Walk the module list to second entry and find the pointer to kernel32.dll's address.
5th: Write duplication of kernel32.dll's address to the pointer

At this point, ShellCodes can only get the address of our duplication. However, we still need to prevent APIs in our kernel32.dll to be found by ShellCodes. To do this, we searched the ExportTable in our duplication and rewrote the address of "LoadLibraryA" and "GetProcAddress." This way, we can monitor the utilization of both addresses and stop the execution of ShellCodes.

3.3 API Hooking and Monitoring

Windows systems create TEB for each thread and create PEB for each process. This means that we need to modify all the TEBs and PEBs when processes and threads are created. API hooking technique therefore achieves this goal. By directly patching the entry point of each Win32 API, we can inject our method into the selected API calls [1, 2, 12, 13].

After mapped our own function which is placed in the hooking DLL to all processes' memory by Win32 API CreateRemoteThread()[13], we rewrote each runtime processes' IAT Table which contains API addresses assigned by the Loader to our own functions' addresses in hooking DLL which just mapped into process memory and recorded its original address of IAT in an array (we can recovery it if unhook our DLL). When process calling the API we rewrote in IAT, it will jump to our functions first. Once this happens, we can then run our methods as processes call it. After our methods have finished, we would call the real Win32 API functions to do the work that process wanted to do and return the function.

Fig. 2 shows the API call sequence of a runtime processes by applying our Win32 API hooking technique. The process runs our method before calling APIs within the kernel32.dll. Our method changes the addresses and makes ShellCodes call APIs within our own DLL instead of the original one. Therefore, we can tell if APIs are called by normal processes or ShellCodes. Our API hooking and monitoring system does not connect to internet directly, so it's hard to inject ShellCodes to API Monitoring system.

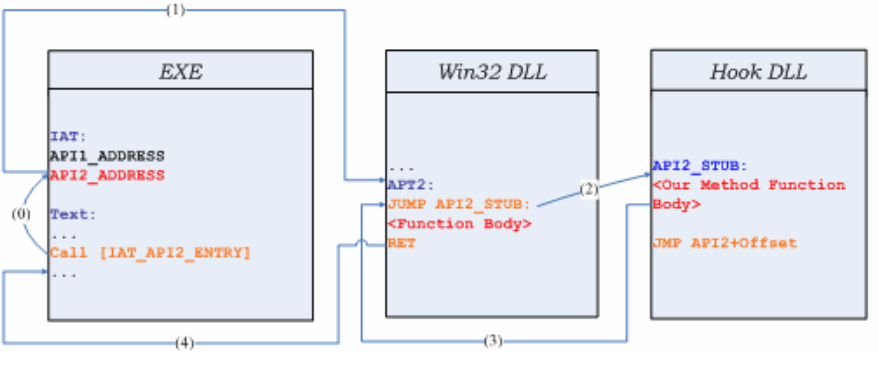


Fig. 2. API Hooking Method. First, the executable calls an API, API2, described in its IAT table. Then it links to the target Win32 DLL. However, we change the address of API2 in DLL file. Therefore, the ShellCodes will get the wrong address about API2 since the address points to our Hook DLL. But for normal processes, they can still call the API2 function as they wish.

3.4 Rebasing Kernel32.dll

For the remaining two types of ShellCodes, our solution uses rebasing kernel32 DLL to protect them. According to our study, every image file, DLL or executable, has an "ImageBase." This base is the preferred location where the file should be loaded into memory by the loader in windows system. If the loader can give the preferred location to image file then there is no need for relocating the image file. Rebasing is a process of changing the ImageBase [3]of an image file. By rebasing process the DLL/EXE file is loaded into a biased location in the memory space, the ShellCodes would never be success since rebasing changes the addresses of API calls. Changing the ImageBase of a specific DLL/EXE file, we have to modify the optional header and relocation field in the PE32 file [13-15].

After we finished rebasing kernel32 DLL, the last problem is the file protection mechanism. We need to replace the new kernel32.dll with the old one, but file protection will not allow users to do this. The solution is to use the API function MoveFileEx() and specify the MOVEFILE_DELAY_UNTIL_REBOOT flag. By doing this, a registry value, "PendingFileRenameOperations," will be added to HKLM\System\CurrentControlSet\Control\Session Manager\. From there, another DWORD value "AllowProtectedRenames" has to be added and set to 1 before we reboot the system. While rebooting, the new kernel32.dll along with its new image bases will be loaded.

4 Security and Performance Analysis

In this section, we give the security analysis and performance evaluation of our scheme. The test platform we selected is the Windows XP Professional Service Pack 1 since there were many exploits for our testing.

4.1 Security Analysis

In this section, we tested each part of our method by three types of ShellCode which were mentioned in Section 2.2 and JMP ESP attack [17]. JMP ESP attack is a technique to locate the ShellCode address in stack memory indirectly and execute it. It can attack successfully the "Variations of Stack Memory Address" and "Change the location of Software Process" schemes. However, it failed in out total solution and rebasing method.

Table 1 illustrates the results of the testing. We can see our mechanism can stop all types of ShellCode.

4.2 Performance Analysis

In this section we combined three parts: rebasing Kernel32.dll, our own PE loader and Win32 API Hooking into our total solution. Generally, Rebasing Kernel32 DLL method affects the performance only when the system startups. The PE loader method affects the performance when the processes are loaded in to memory. Lastly, the Win32 API Calls Hooking affects the performance when some interceptions occur.

[3] ImageBase is the base address of a specific file when it has been mapped into the memory.

Table 1. Security Analysis Using Different Attacking

Method	Manual Input Addresses of API Calls	Scan Memory to Find API Addresses	Obtained API addresses from PEB and TEB	JMP ESP Attack
CLSP	Success	Success	Success	Success
VSMA	Failure-Can not return to the location of ShellCode	Failure-Can not return to the location of ShellCode	Failure-Can not return to the location of ShellCode	Success
Rebasing	Failure-Input wrong API addresses	Failure-Can not find API addresses	Success	Failure-Can not return to ShellCode
Modify TEB and PEB by API hooking	Success	Success	Failure-Can not find API addresses	Success
Total Solution	Failure-Input wrong API addresses	Failure-Can not find API addresses	Failure-Can not find API addresses	Failure-Can not return to ShellCode

After experience, our total solution decreases 8%~9% performance compared to original system when it applied. In addition, the windows system also delayed 0.5 second when it startups.

5 Related Works and Comparison

Methods of anti buffer overflow can be classified into two major categories, Misuse Detection and Anomaly Detection. Most of antivirus and internet security software focus on the first type. They detect the known virus or worms by identifying their signatures. Conversely, anomaly detection methods detect malicious codes through their behaviors that were extracted before.

Several anomaly detection techniques which use stack execution prevention have been proposed to defeat malicious codes. “Microsoft Data Execution Prevention” [20] is the most famous one. It marks the addresses above 0xbfffffff where the stacks are non-executable. If malicious codes try to execute in stack, the operating system will terminate the process. However, this method needs 64-bit processor to support it such as AMD 64-bit processor.

DOME [4] and Behavior Modeling systems both use the system call monitoring and as a result, they both require preprocess steps. Behavior Modeling systems do system call tracing as their preprocess steps and create models of normal behaviors. In stead of tracing system calls(such as Win32 APIs in Windows system), DOME use IDA Pro [8] to disassemble Software executions as preprocess step and identify all Win32 APIs that the process will use. And monitoring all Win32 APIs, see if there are malicious codes which use APIs not belong to the original software process.

The limitation of system call tracing at runtime is that models created during its learning phase would only learn the behavior that can be observed. Unlearned behaviors could cause false-positive during anomaly detection. Unlike tracing system calls at runtime, DOME uses static analysis (disassemble) of binary code to identify Win32 APIs of the software processes. Nevertheless, the “IDA Pro” that DOME used

as the dis-assembler also has its limitation. Not only it takes too much time to disassemble executable files but it also cannot handle dynamic load library and encrypted executable files. Some features might even cause false-positives to occur at this step. The comparisons of other existing anomaly detection techniques on Windows system with our mechanism is given in the table 2.

Our mechanism is an anomaly detection technique. It separates the behaviors of ShellCodes from the normal software processes and stops them at the injection step or runtime step. However, our mechanism does not need additional hardware support. Moreover, we do not even modify the operating system or the underlying instruction set in our mechanism [6]. The most important of all, our mechanism does not suffer false positive or false negative problems since the main trick of our mechanism is hiding or redirecting the original base addresses of APIs.

In table 2, Microsoft Data Execution Prevention has the best performance. But it needs a CPU with special feature as support and it only works on Windows XP Service Pack 2 (might work on Windows 2003 Server Service Pack 1 in the future). Our method has similar or even better performance with behavior modeling techniques which also need techniques like Win32 API Calls Hooking to monitor and construct behavior model and better performance than DOME. In the steps of preprocess, behavior modeling techniques have to run their modeling functions which takes much longer time than running disassembly preprocess works of DOME.

Table 2. Total Comparisons with Other Schemes

Anti-Buffer-Overflow System	DOME	MDEP	Our Method	BMS
Hardware Support	N	64bit-processor	N	N
Runtime Performance	5% slower per API call	Highest	8% slower of all	High
Source Code of Software	N	N	N	N
Working On Encrypted Binary Executables	N	Y	Y	Y
Working Platform	XP/2000	XP SP2	XP/2000	XP/2000
Preprocess Step	Y	N	N	Y
Stop Unknown Malicious Code	Y	Y	Y	Y
Have False Positives	Y	N	N	Y
Stop attacks and intrusion through heap overflow	Y	N	Y	Y

This table shows the cons and pros with ours and the related research. MDEP is Microsoft Data Execution Prevention system. BMS is Behavior Modeling System. MDEP can only apply on Windows XP with Service Pack 2 system.

6 Conclusion

Here we present a scheme for defeating malicious codes that have been injected into runtime software processes. The idea of our scheme is preventing ShellCodes from being injected into right position in the memory and getting memory address of Win32 API Calls. By replacing the rebased kernel32.dll and using Win32 API Calls Hooking to runtime processes, we have successfully defeated the injected malicious

codes that were placed into runtime software processes by worms and exploits through buffer overflow or heap overflow attacks.

Our solution is more efficient than others, and it does not need additional hardware support. In the future, our mechanism will be more efficient by replacing the I/O function by MAPI functions in the API hooking work. Furthermore, better performance will be obtained by rewriting our mechanism as the system service.

Acknowledgements

The authors wish to acknowledge the anonymous reviewers for valuable comments. This research was supported in part by the National Science Council, Taiwan, under contract NSC-94-2213-E-007-039.

References

1. B.Anton, “Process-wide API spying - an ultimate hack,” CodeProject website. Available: http://www.codeproject.com/system/api_spying_hack.asp , Feb 2005.
2. B.Anton, “Kernel-mode API spying - an ultimate hack,” CodeProject website. Available: http://www.codeproject.com/system/kernelspying.asp , Feb 2005.
3. B.Michel, “Introduction to Shellcoding - How to exploit buffer overflows, “ Tigerteam’s website. Available: http://tigerteam.se/dl/papers/intro_to_shellcoding.pdf , 2004.
4. C.Jesse, R.Rabek. I.Khazan, M.Scott, L.Robert and K.Cunningham, “Detection of Injected, Dynamically Generated,and Obfuscated Malicious Code” In Proc. of 2003 ACM workshop on Rapid Malcode October 2003.
5. C.Shannon and D.Moore. “The spread of the Witty worm , ” In Security & Privacy Magazine, IEEE Volume 2, Issue 4, pp. 46 – 50 ,July-Aug. 2004.
6. E. G. Barrantes, D. H. Ackley, T. S. Palmer, D. Stefanovic, and D. D. Zovi. “Randomized instruction set emulation to disrupt binary code injection attacks,” In Proc. of 10th ACM Conf. Comp. and Comm.Sec.—CCS 2003, pp. 281–289. ACM Press, Oct. 2003.
7. E.Levy. “Worm propagation and generic attacks, ” In Security & Privacy Magazine, IEEE Volume 3, Issue 2, pp. 63 – 65, Mar-Apr 2005.
8. G.Hunt and D.Brubacher(1999), “Detours: Binary Interception of Win32 Functions, “ Microsoft corp research website. Available: ftp://ftp.research.microsoft.com/pub/tr/tr-98-33.pdf
9. I.Ivo, “API hooking revealed,” CodeProject website, Available: http:// www.codeproject.com/system/hooksys.asp , Feb 2005.
10. J.Richter, “Programming Applications for Microsoft Windows 4th Edition,” 2001.
11. J.Riordan, A.Wespi and D.Zamboni, “How To Hook Worms, ” In Spectrum, IEEE Volume 42, Issue 5, pp. 32 – 36, May 2005.
12. J.Pincus and R.Baker, “Beyond stack smashing: recent advances in exploiting buffer overruns, ” In Security & Privacy Magazine, IEEE Volume 2, Issue 4, pp. 20 – 27, July-Aug. 2004.
13. M.Pietrek, “Inside Windows: An In-Depth Look into the Win32 Portable Executable File Format,” MSDN Website. Available: http://www.msdn.microsoft.co , 2002.
14. R.S.Sachin, “Need for Rebasing a DLL,” Code Project website. Available: http://www.codeproject.com/dll/RebaseDll.asp, Mar 2005.
15. R.S.Sachin, “Need for Binding an Executable to DLLs,” Code Project website. Available: http://www.thecodeproject.com/dll/NeedBind.asp , Mar 2005.

16. Udo Payer, Peter Teufl and Mario Lamberger, “Hybrid Engine for Polymorphic Shellcode Detection,” In Second Int. Conf. of 2005 Intrusion and Malware Detection and Vulnerability Assessment, July, 2005.
17. Y.Kaplan, “API Spying Techniques for Windows 9x, NT and 2000,” From website of teaching API Hooking and Monitoring. Available:http://www.internals.com/articles/apispy/apispy.htm, June 2004.
18. The MetaSploit Project, “ShellCode Archive,” MetaSploit Project official website. Available:http://www.metasploit.com/ShellCode.html, Nov 2004.
19. Microsoft Corp, “A detailed description of the Data Execution Prevention (DEP) feature in Windows XP Service Pack 2 and Windows XP Tablet PC Edition 2005, ” Microsoft Corp’s support website. Available: http://support.microsoft.com/kb/875352/en-us, Feb 2005.
20. The NTInternals.net team, (Nov 28 2004) “Undocumented Functions for Microsoft Windows NT/2000,” NTInternals.net website. Available: http://undocumented.ntinternals.net , Nov 2004.
21. Phrack Inc, “History and Advances in Windows ShellCode,“ In Phrack Magazine. Available: http://www.phrack.org/phrack/62/p62-0x07_Advances_in_Windows_ShellCode.tx , Nov 2004.
22. Smiler, “The Art of Writing ShellCode,” FreeGnu’s personal blog. Available: http://blog.codelphi.com/freegnu/archive/2004/11/25/29682.aspx , June 2004.
23. The ShellCode.org, “The ShellCode Writing,” ShellCode.org website. Available:http://ShellCode.org/, Nov 2004.

Hiding Circuit Topology from Unbounded Reverse Engineers

Yu Yu, Jussipekka Leiwo, and Benjamin Premkumar

Nanyang Technological University, School of Computer Engineering, Singapore
yuyu@pmail.ntu.edu.sg, {ASJLeiwo, ASANNAMALAI}@ntu.edu.sg

Abstract. Circuit/program obfuscation, if possible, would have a number of applications to cryptography and software protection. Unfortunately, negative results have been given by Barak et al. [1] that universal obfuscators (for circuits or programs) do not exist. In other words, given a circuit, an adversary might obtain more information (e.g. core algorithm) other than its input-output behavior. In this paper, we discuss the problem of circuit obfuscation under a weaker assumption where the adversary knows only partial information regarding the circuit, namely, the circuit topology (i.e., all information regarding the circuit except the functionalities of its gates), then how can C be obfuscated such that the circuit topology of the resulting circuit C' (denoted by $Topo(C')$) discloses nothing substantial? In practice, the scenario corresponds to that a reverse engineer attempts to illegally copy a circuit by passively analyzing how its gates are inter-wired. Our results are quite positive: there exist efficient circuit topology obfuscation algorithms that transform every circuit C with size s to circuit C' with the same input-output behavior, size $s \cdot \log^3 s$ and depth $s \cdot \log(\log s)$, where $Topo(C')$ reveals nothing more than circuit size, input length and output length in an information-theoretic sense.

1 Introduction

1.1 Background

Informally, obfuscation refers to an efficient transformation of software programs or circuits aiming at disclosing nothing but their input-output behaviors (i.e. like an oracle access). If such a universal obfuscation algorithm exists, it will have a number of applications to software protection and cryptography. For example, software can be obfuscated such that the core algorithm cannot be extracted by reverse engineers; software developers can watermark (i.e. by obfuscating a program that is embedded in an ownership code) each copy of their software uniquely so as to trace unauthorized redistribution (e.g., [1, 2]); new public-key encryption schemes can be obtained by obfuscating existing private-key encryption schemes with private keys embedded in [3]. Unfortunately, Barak et al. [4] have proved that universal obfuscation algorithms for programs or circuits do not exist, where by "universal" they do not rule out the fact that some classes of problems are obfuscable under cryptographic assumptions [5].

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 171–182, 2006.
© Springer-Verlag Berlin Heidelberg 2006

1.2 Motivation

In this paper, we seek to find a practical scenario where obfuscation is possible. We assume that a computationally unbounded adversary knows the circuit topology (see the formal definition in Sect 2.1) of C and he attempts to recover C. In practice, it is usually hard to hide [1] the skeleton of a circuit from a reverse engineer. Furthermore, circuit topology may carry non-trivial information regarding the circuit due to the following reasons:

1. The basis (i.e., set of gate functionalities) of C usually has a small number of elements, e.g., OR gate ($\vee$), AND gate ($\wedge$) and NOT gate ($\neg$), with each element having a specific range of fan-in, e.g., given the topology of a circuit with basis $\{\vee, \neg\}$, it is a trivial task to recover the circuit by assigning gate nodes of fan-in 1 with $\neg$ and the rest with $\vee$.
2. For the sake of efficiency, C is usually designed in a way such that all its subcircuits computing fundamental functions (e.g., addition, multiplication, matrix product) are in the minimized format and an adversary can easily recognize them from their topology, e.g., we can easily identify the circuit of a full-adder given only how the circuit is wired.

Thus, in some cases, an adversary could (partially) recover C given its circuit topology. It is necessary for us to obfuscate C to make its topology reveal nothing substantial. Circuit topology obfuscation can be considered as an efficient algorithm that transforms C to C' such that the following conditions hold:

1. (functionality): C' has the same input-output behavior as C does, i.e., for any valid input x, it holds that $C(x)$=$C'(x)$.
2. (overhead): Compared with C, C' is increased by only a reasonable factor (e.g., poly-logarithmic) in circuit size.
3. (privacy): The circuit topology of C' (denoted by $Topo(C')$) reveals nothing more than the input length, output length and circuit size.

We note that there is a trivial alternative for circuit topology obfuscation. That is, we can construct a universal circuit U [9] such that for any valid input x, $U(C, x) = C\ (x)$. Due to the universal property of U, $Topo(U)$ reveals nothing more than the size of C and the length of x. Nevertheless, U is not an obfuscation of C since it takes the encoding of C as an input (i.e., not satisfying condition 1) and hence the input of U may be significantly long. Furthermore, even if we hardwire C in U, such an obfuscation is not efficient enough because to simulate C with size s and depth d, U should have size $O(d{\cdot}s{\cdot}\log s)$.

[1] The circuit topology is hard to hide even in case of secure computation protocols. Consider a secure two-party computation [6], where a circuit is encrypted (refer to [7, 8] for detailed encryption techniques) such that the encrypted circuit is computed by a party without revealing the computation transcript (i.e. inputs, intermediate results and outputs). However, the party can still know the circuit topology since it is a necessary knowledge for evaluating the circuit.

1.3 Our Contribution and Related Works

In this paper, we propose an efficient circuit topology obfuscation algorithm that transforms any circuit C with size s to a C' with the same input-output behavior, size $s{\cdot}\log^3 s$ and depth $s{\cdot}\log(\log s)$. More importantly, we prove that $Topo(C')$ reveals nothing more than its input length, output length and size. Therefore, by replacing C with C', we obtain the same functionality with only a poly-logarithmic cost and C' is unconditionally secure against adversaries analyzing its topology.

2 Preliminaries to Boolean Circuits

2.1 Assumptions and Notations

A circuit C with m outputs can be viewed as m subcircuits (or Boolean functions) $C_1, \cdots, C_m$, where each C_i $(1{\le}i{\le}m)$ has only 1 output that coincides with the i-th output of C. Thus, for simplicity, we assume hereafter that C has n inputs, Γ gates, 1 output and fan-in 2 and that the functionalities of gates are defined over the basis

$$\Omega_2 = \{g \mid g : \{0,1\} \times \{0,1\} \to \{0,1\}\} \ .$$

In other words, g can be either a basic gate (e.g. $g(a,b){=}a{\vee}b$), a non-trivial gate (e.g. $g(a,b){=}a{\wedge}\bar{b}$) or even a degenerate gate (e.g. $g(a,b){=}a$), where a degenerate gate takes as inputs two wires and outputs one of their results or a Boolean constant (i.e. neither of the inputs), and a non-trivial gate is internally made of several basic gates. We note that NOT gate (gate of fan-in 1) is not necessary as it can be integrated into its adjacent gate, e.g., $g(a,b)$ followed by a NOT gate can be simplified to $g'(a,b){=}\ g(a,b){\oplus}1$. Therefore, the encoding of C can be written as follows:

$$\begin{aligned} &v_1, \cdots, v_n \\ &v_{n+1} = g_{n+1}(v_{a_1}, v_{b_1}) \\ &\vdots \\ &v_{n+\Gamma} = g_{n+\Gamma}(v_{a_\Gamma}, v_{b_\Gamma}) \end{aligned}$$

where $v_1, \cdots, v_n$ are inputs, $v_{n+1}, \cdots, v_{n+\Gamma-1}$ are intermediate results and $v_{n+\Gamma}$ is the final output, and $a_i < b_i < n+i$ for all $1 \le i \le \Gamma$. The circuit topology of C, $Topo(C)$, is defined as

$$(n, \Gamma, 1), (a_1, b_1), (a_2, b_2), \cdots, (a_\Gamma, b_\Gamma)$$

2.2 Oblivious Permutation Circuits

Before presenting the circuit obfuscation algorithm, we introduce how to construct a subcircuit C_π capable of permuting the results of a list of nodes obliviously. As shown in Fig. 1.(A), we suppose that the inputs of the n_1 gates numbered $2n_1{+}1, \cdots, 3n_1$ come from the outputs of the $2n_1$ nodes numbered $1, \cdots,$

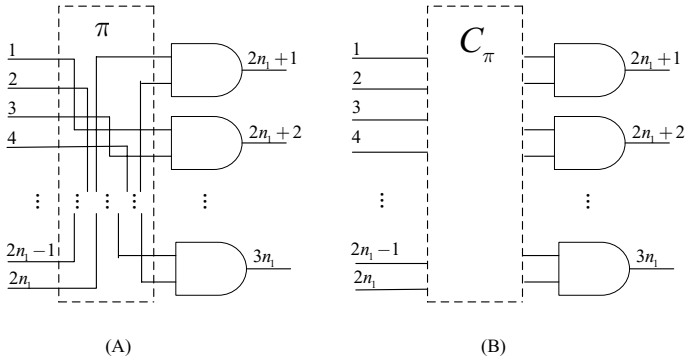


Fig. 1. (A) The outputs of nodes numbered 1, $\cdots$, $2n_1$ are permuted according to permutation function π:{1, $\cdots$, $2n_1$}$\rightarrow${1, $\cdots$, $2n_1$} before they are used by gates numbered $2n_1$+1, $\cdots$, $3n_1$. (B) A subcircuit C_π that is functionally equivalent to π while $Topo(C_\pi)$ is independent of π.

$2n_1$. At this point, we also assume that there is a one-to-one correspondence between the inputs and outputs, namely, each result of node i ($1 \leq i \leq 2n_1$) is used exactly once. It is trivial for anyone observing the circuit topology to tell which nodes contribute to the inputs of gate node j ($2n_1 < j \leq 3n_1$). Thus, we need to find an efficient subcircuit construction algorithm such that:

- Given a permutation π:{1,$\cdots$,$2n_1$}$\rightarrow${1,$\cdots$,$2n_1$}, it efficiently outputs a $2n_1$-input-$2n_1$-output subcircuit C_π, where the $\pi(i)$-th output carries the result of the i-th input for all $i \in$ {1,$\cdots$,$2n_1$}.
- $Topo(C_\pi)$ is uniform regardless of π (i.e. only depends on the value of $2n_1$).

After we obtain C_π, we insert it in between the circuit as shown in Fig. 1.(B), then the topology of the resulting circuit reveals nothing about the inputs of each gate j ($2n_1 < j \leq 3n_1$).

We reduce the problem of generating C_π to a sorting network problem: There are $2n_1$ variables, x_1, x_2, $\cdots$, x_{2n_1}, which are initialized to $\pi(1)$, $\pi(2)$, $\cdots$, $\pi(2n_1)$ respectively. Each x_i ($1 \leq i \leq 2n_1$) is attached to node l_i with l_i initialized to i. A comparison between x_i and x_j involves the following actions:

1. Suppose that the current maximum node number is L, then we generate two gates $g_{L+1}(v_{l_i}, v_{l_j})$ and $g_{L+2}(v_{l_i}, v_{l_j})$, where v_{l_i} and v_{l_j} denote the outputs of node l_i and node l_j respectively and

$$g_{L+1}(v_{l_i}, v_{l_j}) = \begin{cases} v_{l_i}, \text{ if } x_i < x_j \\ v_{l_j}, \text{ otherwise} \end{cases} \qquad g_{L+2}(v_{l_i}, v_{l_j}) = \begin{cases} v_{l_j}, \text{ if } x_i < x_j \\ v_{l_i}, \text{ otherwise} \end{cases}$$

2. Swap the values of x_i and x_j if $x_i > x_j$. Re-attach x_i and x_j to node L+1 and node L+2 respectively (i.e. $l_i \leftarrow L$+1 and $l_j \leftarrow L$+2) and increment L by 2.

How can the list $x_1, x_2, \cdots, x_{2n_1}$ be sorted to an ascending order by performing comparisons in a predetermined manner (i.e. data independent)? When the list is sorted, C_π is produced as well: the input nodes of C_π are node $1, \cdots,$ node $2n_1$, gates of C_π are generated during comparisons and output nodes are numbered $l_1, \cdots, l_{2n_1}$ with $v_{l_{\pi(i)}}=v_i$ $(1\leq i\leq 2n_1)$.

It is easy to verify that the above reduction is correct. Firstly, during each comparison, v_{l_i} and v_{l_j} are swapped (by generating $g_{L+1}(v_{l_i},v_{l_j})=v_{l_j}$, $g_{L+2}(v_{l_i},v_{l_j}) = v_{l_i}$ and letting $l_i=L+1$ and $l_j=L+2$) if and only if x_i and x_j are swapped. Initially, it holds that $x_i=\pi(i)$ and $v_{l_i}=v_i$. By the time $x_1, \cdots, x_{2n_1}$ are sorted, it will hold that $x_{\pi(i)}=\pi(i)$ and hence the $\pi(i)$-th output of the permutation subcircuit $v_{l_{\pi(i)}}$ is equal to v_i. Secondly, no matter $x_i < x_j$ or not, two gates are generated in the same way in terms of circuit topology. Finally, the comparisons are performed in an order regardless of π and hence the resulting circuit topology is uniform for all $2n_1$-to-$2n_1$ permutations.

A satisfactory [2] solution for the reduced problem is given by Batcher [11], who uses $O(n\cdot\log^2 n)$ comparisons with a depth of $O(\log^2 n)$ to sort n elements in a data-independent way. Since each comparison generates two gates, we can construct an n_1-input-$O(n_1\cdot\log^2 n_1)$-gate-n_1-output permutation subcircuit C_π with depth $O(\log^2 n)$ using a uniform topology.

In some cases, we also hope to produce an n_1-input-n_2-output subcircuit with $n_1<n_2$ and the input-output mapping is an injection, namely, there is a bijection π_1:$\{1, \cdots, n_1\} \rightarrow \{a_1, \cdots, a_{n_1}\}$ with $\{a_1, \cdots, a_{n_1}\} \subset \{1, \cdots, n_2\}$ and the $\pi_1(i)$-th output equal to the i-th input. In this case, we construct another arbitrary bijection π_2: $\{n_1+1, \cdots, n_2\} \rightarrow (\{1, \cdots, n_2\}-\{a_1, \cdots, a_{n_1}\})$ and let permutation $\pi= \pi_1 \cup \pi_2$. We append n_1-n_2 constant inputs (e.g. 0) to the end of the input list and proceed to generating C_π using according to π. During the comparison, if a resulting gate has one or two constant inputs (i.e. a degenerate gate), e.g., $g_k(v_i,0)$ or $g_k(0,0)$, write it as $g_k(v_i,v_*)$ or $g_k(v_*,v_\#)$, where $*$ and $\#$ can be arbitrarily chosen. Thus, by padding (n_2-n_1) constant inputs, we obtain C_π capable of permuting n_1 inputs into n_2 outputs obliviously.

2.3 Oblivious Multiplexer Circuits

Another useful subcircuit for obfuscation is the multiplexer circuit C_{mux} that takes as input the results of several nodes (i.e., $v_1, \cdots, v_{n_3}$) and only selects one of them as output (i.e., v_s). As shown in Fig. 2, the topology of C_{mux} is a balanced binary tree, where leaf nodes take as inputs v_1 through v_{n_3}, root node outputs v_s, and each level holds the maximal number of nodes except that the highest level keeps its nodes aligned to the left. There will be a connected path (e.g., see the highlighted path in Fig. 2) from wire v_s (at the leaf node level) to the output wire of the root node. We assign assign degenerate functionalities to those on-path gates (e.g., nodes numbered 1, 2, 4, 9, 18 in Fig. 2) such that the output is equal to the on-path incoming

[2] There is an asymptotically better result (also the lower bound), namely, the AKS network [10] with $O(n\cdot\log n)$ comparisons and depth $O(\log n)$, but it is impractical as the constant factor hidden by the big-oh notation is extremely large.

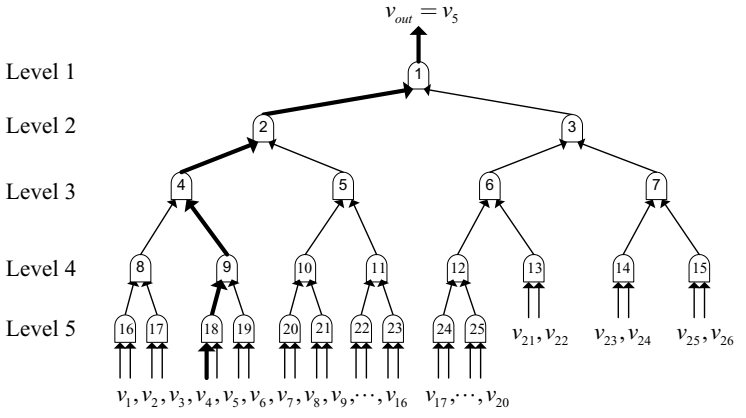


Fig. 2. An example of multiplexer sub-circuit C_{mux} with n_3=26 and selection decision s=5

wire (e.g., the output of gate node 1 is equal to its left child), and assign arbitrary functionalities to the rest gates. As $Topo(C_{mux})$ is a balanced binary tree and is fully determined (recall that we require the nodes on the highest level to be aligned to the left) by the number of inputs n_3, $Topo(C_{mux})$ is independent of s, and C_{mux} has (n_3-1) gates and depth $\lceil \log_2 n_3 \rceil$. We note that the multiplexer circuit C_{mux} introduced here differs to the one in logic design in that the selection s is hardwired in C_{mux} instead of being an input.

3 Circuit Topology Obfuscation

3.1 Obfuscation Algorithm

We start with an overview of the circuit obfuscation algorithm $\mathcal{O}$. On input of an n-input-Γ-gate-1-output circuit C, $\mathcal{O}$ obfuscates it gate by gate (i.e. from $n+1$ to $n+\Gamma$). As depicted in Fig. 3, $\mathcal{O}$ maintains up to d permutation subcircuits $C_\pi^1, \cdots, C_\pi^d$ and they are updated before each gate is obfuscated. For each gate $g_{n+j}(v_{a_j}, v_{b_j})$, we obfuscate it by replacing it with $g'_{n+j}(v'_{a_j}, v'_{b_j})$ where g'_{n+j} has the same functionality as g_{n+j} does, and v'_{a_j} and v'_{b_j} are the outputs of two multiplexer circuits $C_{mux}^{a_j}$ and $C_{mux}^{b_j}$ respectively. If we denote by $c_{i,k}$ the k-th output of C_π^i, then the set of inputs of $C_{mux}^{a_j}$ and that of $C_{mux}^{b_j}$ are

$$\{c_{i,k}, \text{ where } 1 \le i \le d,\ t_i = 1 \text{ (switched on) and } k = ((2j-2)\ mod\ 2^i)+1\} \quad (1)$$

and

$$\{c_{i,k}, \text{ where } 1 \le i \le d,\ t_i = 1 \text{ (switched on) and } k = ((2j-2)\ mod\ 2^i)+2\} \quad (2)$$

respectively, where $(2j-2) mod\ 2^i$ means that $(2j-2)$ wraps around after it reaches 2^i-1, namely, only the first 2^i outputs are used by multiplexer circuits.

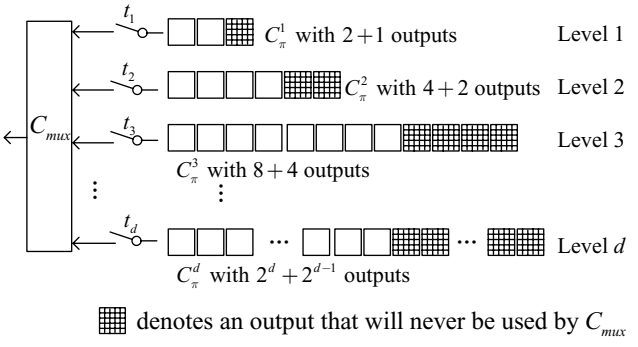


Fig. 3. An obfuscation network that consists of a d-input multiplexer C_{mux} and d permutation subcircuits $C_\pi^1, \cdots, C_\pi^d$, where each C_π^i $(1 \le i \le d)$ has $2^i + 2^{i-1}$ outputs and one of the first 2^i outputs of each C_π^i contributes to the an input of C_{mux} provided that switch t_i is on

Multiplexer subcircuits $C_{mux}^{a_j}$ and $C_{mux}^{b_j}$ are non-reusable, namely, $\mathcal{O}$ generates a distinctive pair of $(C_{mux}^{a_j}, C_{mux}^{b_j})$ for each gate $g_{n+j}(v_{a_j}, v_{b_j})$. In contrast, the d permutation subcircuits C_π^1 through C_π^d are lasting and they will be updated before and after obfuscating each gate. That is, each C_π^i should be viewed as a pointer to a $3{\cdot}2^{i-1}$-input-$3{\cdot}2^{i-1}$-output permutation subcircuit instead of a fixed subcircuit because it will refer to different subcircuits as t_i varies, e.g., C_π^i will no longer refer to a subcircuit when t_i is changed from 1 to 0 and it will point to a new one later when t_i is reset to 1.

Now we introduce the update of the permutation subcircuits. Initially, $d = \lceil \log_2(n) \rceil$ (i.e. $2^{d-1} < n \le 2^d$) and $\mathcal{O}$ produces an n-input-$3{\cdot}2^{d-1}$-output permutation subcircuit C_π^d that takes as inputs v_1 through v_n (the n inputs of C). Since only level d is occupied, $\mathcal{O}$ sets $t_d t_{d-1} \cdots t_1$ (which also serves as a counter with t_1 the least significant bit) to $10 \cdots 0$. Then, $\mathcal{O}$ starts to obfuscate each gate sequentially by replacing $g_{n+j}(v_{a_j}, v_{b_j})$ with $g'_{n+j}(v'_{a_j}, v'_{b_j})$. After each gate g_{n+j} is obfuscated, the d permutation subcircuits are updated as follows:

- If $t_1 = 0$, then $\mathcal{O}$ generates a 3-input-3-output C_π^1 taking as inputs v'_{a_j}, v'_{b_j} and $v'_{n+j} = g'_{n+j}(v'_{a_j}, v'_{b_j})$, increments the counter $t_d t_{d-1} \cdots t_1$ by 1 (i.e. set t_1 to 1).
- If $t_1 = 1$ and there is a b such that $1 < b < d$ and $t_b t_{b-1} \cdots t_1 = 01 \cdots 1$, then $\mathcal{O}$ generates a $3{\cdot}2^{b-1}$-input-$3{\cdot}2^{b-1}$-output C_π^b taking as inputs v'_{a_j}, v'_{b_j}, v'_{n+j} and all the outputs of C_π^1 through C_π^{b-1}, increments the counter $t_d t_{d-1} \cdots t_1$ by 1 (i.e. set t_b to 1 and set $t_{b-1} \cdots t_1$ to $0 \cdots 0$).
- Otherwise, it holds that $t_d \cdots t_1 = 1 \cdots 1$, $\mathcal{O}$ generates a $3{\cdot}2^d$-input-$3{\cdot}2^d$-output C_π^{d+1} taking as inputs v'_{a_j}, v'_{b_j}, v'_{n+j} and all the outputs of C_π^1 through C_π^d, increments the counter by 1 (i.e. d is incremented by 1 and $t_d t_{d-1} \cdots t_1$ is set to $10 \cdots 0$).

$\mathcal{O}$ moves on to the next gate (i.e. increments j by 1) and repeats the above operations until the last gate $g_{n+\Gamma}$ is obfuscated. We can see that each of the first

2^i outputs of C_π^i is used (by multiplexer subcircuits) exactly once (i.e. $t_i t_{i-1} \cdots t_1$ counts from $10\cdots0$ to $11\cdots1$) before C_π^i is pulled out from level i (i.e. t_i is set to 0) and its outputs are re-shuffled (together with those in lower levels) to a higher level. The counter $t_d t_{d-1} \cdots t_1$ is incremented by 1 after each gate obfuscation.

We have described how to generate the circuit topology of obfuscated circuit C'=$\mathcal{O}(C)$. Obviously, $Topo(C')$ only depends on $(n,\ \Gamma,\ 1)$ since all the intermediately generated C_{mux}'s and C_π's are uniform in terms of topology. It only remains to be shown how to determine the functionalities of the gates in C', which fall into two categories : g'_{n+j} that corresponds to g_{n+j} and gates that reside in C_{mux} and C_π. As we have discussed, g'_{n+j} is assigned the same functionality as g_{n+j}, and C_π (resp., C_{mux}) is fully determined by n_1 and π (resp.,

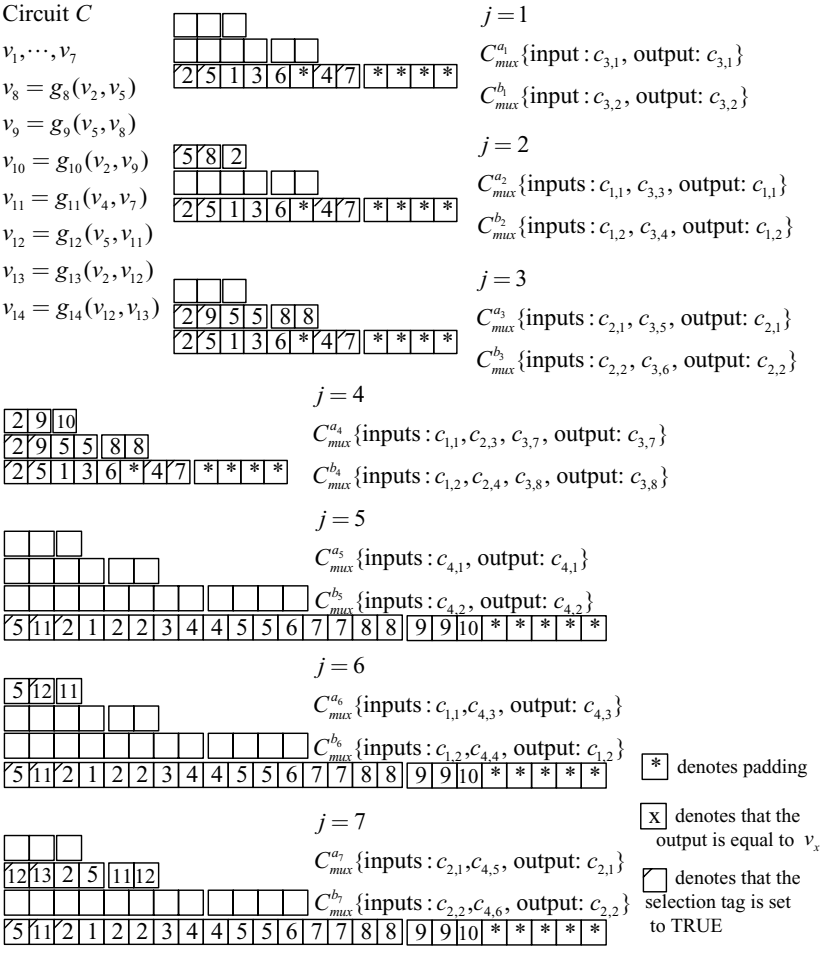


Fig. 4. An example of how to determine permutation function π and selection s in each step

n_3 and s). It suffices to explain how to determine the permutation function π and the selection s of each C_π and C_{mux} such that for each $j \in \{1, \cdots, \Gamma\}$, it holds that $v'_{a_j}{=}v_{a_j}$ and $v'_{b_j}{=}v_{b_j}$. We illustrate how to determine π and s using a simple example in Fig. 4. We suppose that the current (to be obfuscated next) gate number is j and that C^i_π is to permute a set of inputs that correspond to v_{L_1}, v_{L_2}, $\cdots$ $v_{L_{3\cdot 2^{i-1}}}$. The cardinality of the set is less than $3{\cdot}2^{i-1}$ since there may be padded constants and overlapping values in the set. We list out the node numbers of inputs to the next 2^{i-1} gates ($g_{n+j}(v_{a_j},v_{b_j})$ through $g_{n+j+2^{i-1}-1}(v_{a_{j+2^{i-1}-1}},v_{b_{j+2^{i-1}-1}})$):

$$a_j,\ b_j,\ a_{j+1},\ b_{j+1},\ \cdots,\ a_{j+2^{i-1}-1},\ b_{j+2^{i-1}-1} \tag{3}$$

and construct a bijection π_1: $S_1 \rightarrow \pi_1(S_1)$, where $S_1 = \{L_1, \cdots, L_{3\cdot 2^{i-1}}\} \cap \{a_j, b_j, \cdots, a_{j+2^{i-1}-1}, b_{j+2^{i-1}-1}\}$ and $\pi_1(x)$ is the position number of the leftmost x that appears in the list (3) (x may appear more than once). We also define an arbitrary bijection π_2:

$$\{1, \cdots, 3 \cdot 2^{i-1}\} - S_1 \rightarrow \{1, \cdots, 3 \cdot 2^{i-1}\} - \pi_1(S_1)\ .$$

Finally, we get permutation $\pi{=}\pi_1{\cup}\pi_2$ and mark the selection tags of the outputs (of C^i_π) whose positions are given by $\pi_1(S_1)$ as TRUE. The selection s of $C^{a_j}_{mux}$ (resp., $C^{b_j}_{mux}$) is determined by choosing a $c_{i,k}$ from set Eq. 1 (resp., set Eq. 2) whose selection tag is TRUE and letting $s = i$. For example, in Fig. 4, when $j{=}1$, $S_1{=}\{1, \cdots, 7\}{\cap}\{2{,}5{,}5{,}8{,}2{,}9{,}4{,}7\}{=}\{2{,}4{,}5{,}7\}$ and hence $\pi(2)$, $\pi(4)$, $\pi(5)$ and $\pi(7)$ is set to 1, 7, 2 and 8 respectively. It follows that the 1st, 2nd, 7th and 8th outputs of C^3_π are all tagged as TRUE. Thus, the outputs of $C^{a_1}_{mux}$ and $C^{b_1}_{mux}$ are $c_{3,1}$ and $c_{3,2}$ respectively and the selections of $C^{a_1}_{mux}$ and $C^{b_1}_{mux}$ are both 3.

Correctness. Now we discuss that each $C^{a_j}_{mux}$ (resp., $C^{b_j}_{mux}$) can always ensure its output $v'_{a_j}{=}v_{a_j}$ (resp., $v'_{b_j}{=}v_{b_j}$) by identifying the tagged (as TRUE) input out of its input candidates given in Eq. 1 (resp., Eq. 2). Since $a_j{<}b_j$, v_{a_j} and v_{b_j} always refer to outputs of different nodes. During each round when the counter $t_d t_{d-1} \cdots t_1$ counts from $10\cdots0$ (i.e. at initialization or when d is incremented by 1) to $11\cdots1$, $\mathcal{O}$ will obfuscate 2^{d-1} gates (unless there are less than 2^{d-1} gates left) before entering the next round. The node numbers of inputs to these 2^{d-1} gates are

$$a_j,\ b_j,\ a_{j+1},\ b_{j+1},\ \cdots,\ a_{j+2^{d-1}-1},\ b_{j+2^{d-1}-1}$$

Without loss of generality, we consider an arbitrary a_w in the above list.

- If v_{a_w} is found is the inputs of C^d_π (i.e. $a_w < j$).
  - If a_w appears only once in the list or it is the leftmost one among those values that appear in the list and are equal to a_w, then $\pi(a_w)$ is equal to its position number and it is hit in the d-th level (i.e. the corresponding selection s of $C^{a_w}_{mux}$ is d).
  - Otherwise, there is at least a a_u (or b_u) that is equal to a_w and $u{<}w$. Again without loss of generality, we assume that it is a_u. Thus, after a_u

is hit on level d and gate $g_{n+u}(v_{a_u},v_{b_u})$ is obfuscated, v_{a_u} (v_{a_u}=v_{a_w}) will be copied and permutated to a lower level i'. If on level i', a_w is the first among those node numbers (in the corresponding list of level i') whose values are equal to a_w, it is hit on level i'. Otherwise, the operation is repeated recursively until all $g_{n+u}(a_u,b_u)$'s satisfying $u<w$ and (a_u or b_u)=a_w are obfuscated.

– Otherwise, it holds that $a_w \geq j$ and gate g_{a_w} has not been obfuscated yet. We wait until g_{a_w} is obfuscated and v_{a_w} is copied to a lower level i'. Analogously, if in the corresponding list of level i' there is no such a_u or b_u that is equal to a_w and $u<w$, a_w will be hit on level i', otherwise, the recursion continues until a_w is hit.

3.2 Information Disclosure of $Topo(C')$ and Obfuscation Overhead

We have finished the description of algorithm $\mathcal{O}$ as well as its correctness (i.e. C is functionally equivalent to $\mathcal{O}(C)$). We find that the topology of the obfuscated circuit is generated in a way depending on only n, Γ and 1. We stress that the permutation function π and selection s have no effect on $Topo(C_\pi)$ and $Topo(C_{mux})$. In addition, the two inputs of gate g'_{n+j} are also fully determined by n, $t_d \cdots t_1$ and the current gate index j (see Eq.(1) and Eq.(2)), where j is incremented sequentially and $j-(t_d \cdots t_1)$ is a constant. Thus, $Topo(\mathcal{O}(C))$ is independent of the values of

$$(a_1, b_1), (a_2, b_2), \cdots, (a_\Gamma, b_\Gamma)$$

and the following theorem holds.

Theorem 1. *(Information disclosure of Topo(C')): Let $\mathcal{O}$ be as in Sect. 3.1 and let C be an n-input-Γ-gate-1-output bounded fan-in Boolean circuit, then on input C, $\mathcal{O}$ outputs a bounded fan-in circuit C'=$\mathcal{O}(C)$ with $Topo(C')$ revealing nothing more than n, Γ and 1.*

Proof. We give the proof by showing how to obtain $Topo(C')$ efficient using n, Γ and 1. We construct an arbitrary n-input-Γ-gate-1-output circuit $C1$. Then we obfuscate $C1$ using $\mathcal{O}$ to produce $C1'$=$\mathcal{O}(C1)$. As we have discussed, $Topo(C1')$ should be identical to $Topo(C')$ and it follows that $Topo(C')$ discloses no more than n, Γ and 1. □

Theorem 2. *(Obfuscation overhead): Let $\mathcal{O}$ be as in Sect. 3.1 and let C be an n-input-Γ-gate bounded fan-in circuit, then on input C, $\mathcal{O}$ outputs a bounded fan-in circuit C' with size $\Gamma \cdot O(\log^3 \Gamma)$ and depth $\Gamma \cdot O(\log\log \Gamma)$.*

Proof. As we have discussed, the obfuscation process can be considered as working in rounds. d is initialized to $\lceil \log_2(n) \rceil$ and each round starts with $t_d t_{d-1} \cdots t_1 = 10 \cdots 0$ and ends when it reaches $11 \cdots 1$ and d is incremented by 1. During each round, $\mathcal{O}$ obfuscates 2^{d-1} gates (unless it reaches the last gate) and generates 1 instance of C_π^d, 2^0 instance of C_π^{d-1}, 2^1 instances of C_π^{d-2}, $\cdots$, 2^{d-1-i}

instances of C_π^i, $\cdots$, 2^{d-2} instances of C_π^1 and $2\binom{d-1}{0}$ instances of 1-input C_{mux}, $2\binom{d-1}{1}$ instances of 2-input C_{mux}, $\cdots$, $2\binom{d-1}{d-1}$ instances of d-input C_{mux} and g'_{n+j}, $\cdots$, $g'_{n+j+2^{d-1}-1}$. Thus, the total number of gates generated during this round is

$$\begin{aligned}
&O(3\cdot 2^{d-1}\cdot(\log_2 3\cdot 2^{d-1})^2)+O(1)\sum_{i=1}^{d-1}2^{d-1-i}\cdot 3\cdot 2^{i-1}\cdot(\log_2(3\cdot 2^{i-1}))^2\\
&+2\sum_{k=0}^{d-1}k\binom{d-1}{k}+2^{d-1}\\
&=O(d^2 2^d)+O(2^d)\sum_{i=1}^{d-1}i^2+(d-1)2^{d-1}+2^{d-1}\\
&=O(d^3 2^d)
\end{aligned}$$

and the circuit depth accumulated in this round is

$$\begin{aligned}
&O((\log_2 3\cdot 2^{d-1})^2)+\sum_{i=1}^{d-1}2^{d-1-i}O((\log_2 3\cdot 2^{i-1})^2)+\sum_{k=1}^{d}\binom{d-1}{k-1}\log_2 k+2^{d-1}\\
&=O(d^2)+O(1)\sum_{i=1}^{d-1}i^2 2^{d-i}+O(1)\sum_{k=1}^{d}\binom{d-1}{k-1}\log_2 k+2^{d-1}\\
&=O(2^d)+O(2^d\log_2 d)\\
&=O(2^d\log_2 d)
\end{aligned}$$

Suppose that there are R rounds (i.e. $d\in\{\lceil\log_2(n)\rceil,\cdots,\lceil\log_2(n)\rceil+\mathrm{R}-1\}$), then it holds that

$$\sum_{d=\lceil\log_2 n\rceil}^{\lceil\log_2 n\rceil+R-1}2^{d-1}\le\Gamma$$

namely, these R rounds are sufficient to obfuscate Γ gates. It follows that $R{=}O(1)\cdot\log_2(\frac{\Gamma}{n})$ and the number of gates in C' is

$$\sum_{d=\lceil\log_2 n\rceil}^{\lceil\log_2 n\rceil+R-1}O(d^3 2^d)<(\lceil\log_2 n\rceil+R)^3\sum_{d=\lceil\log_2 n\rceil}^{\lceil\log_2 n\rceil+R-1}O(2^d)\le O(\log_2^3\Gamma)\cdot\Gamma$$

and the depth of C' is

$$\sum_{d=\lceil\log_2 n\rceil}^{\lceil\log_2 n\rceil+R-1}O(2^d\log_2 d)<\log_2(\log_2 n{+}R)\sum_{d=\lceil\log_2 n\rceil}^{\lceil\log_2 n\rceil+R-1}O(2^d)=O(\log_2\log_2\Gamma)\cdot\Gamma$$

□

Note that the depth of C' is independent of that of C since $Topo(C')$ is uniform for all C that have the same input length, output length and circuit size.

4 Concluding Remarks

We have shown that a circuit C can be transformed to an obfuscated one C' with the same input-output behavior by paying a poly-logarithmic price in circuit size. In the meantime, $Topo(C')$ reveals nothing more than the input length, output length and circuit size in an information-theoretic sense.

References

1. Grover, D.: Program identification. In: The protection of computer software—its technology and applications. Cambridge University Press (1989) 119–150
2. Naccache, D., Shamir, A., Stern, J.P.: How to copyright a function? In: Proc. 2nd International Workshop on Practice and Theory in Public Key Cryptography (PKC 99), Springer-Verlag (1999) 188–196
3. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **IT-22**(6) (1976) 644–654
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Advances in Cryptology - CRYPTO 2001, Springer-Verlag (2001) 1–18
5. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Advances in Cryptology - EUROCRYPT 2004. (2004) 20–39
6. Yao, A.C.C.: How to generate and exchange secrets. In: Proc. 27rd Annual Symposium on Foundations of Computer Science (FOCS 1986). (1986) 162–167
7. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proc. 19th Annual ACM Symposium on Theory of Computing (STOC 1987). (1987) 218–229
8. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols. In: Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC 1990). (1990) 503–513
9. Valiant, L.G.: Universal circuits (preliminary report). In: Proc. 8th Annual ACM Symposium on Theory of Computing (STOC 1976). (1976) 196–203
10. Ajtai, M., Komlós, J., Szemerédi, E.: An O(n log n) sorting network. In: Proc. 15th Annual ACM Symposium on Theory of Computing (STOC 1983). (1983) 1–9
11. Batcher, K.E.: Sorting networks and their applications. In: Proc. AFIPS Spring Joint Computing Conference. (1968) 307–314

The Role of the Self-Defending Object Concept in Developing Distributed Security-Aware Applications

John W. Holford and William J. Caelli

Information Security Institute
Queensland University of Technology,
Brisbane, Queensland 4001
j.holford, w.caelli@qut.edu.au

Abstract. The Self-Defending Object (SDO) concept extends the current object-oriented programming paradigm to specifically target the peculiar requirements of Security Aware Application (SAA) development. This paper discusses the SDO Distribution Architecture (SDODA) that enables the use of the SDO concept in development of distributed SAAs. Specifically the architecture overcomes the apparent incompatibility between the two programming models considered and the SDO concept that prevented the transfer of SDOs between SAA hosts. To demonstrate the applicability of both the architecture and the SDO concept itself, two versions of a distributed (Java) SAA were developed using orthogonal distributed programming models, the Web services and the Java RMI models. This paper deals with the implementation of the SDODA and the SDO concept in a Web services environment.

The successful use of the architecture demonstrated that the SDO concept can be used to simplify the implementation of application level mandatory access control in distributed SAAs.

Keywords: Computer Security, Access Control, Distributed Computing, Object Oriented Systems.

1 Introduction

The last decade has seen a profound shift in how industry and government agencies manage their information systems. Information systems are now almost universally interconnected via the Internet. In an attempt to gain a strategic advantage over their competitors and to reduce the costs associated with doing business, many organisations are moving beyond the mere interconnection of information systems to the full integration of critical components of their own information system with those of their trading partners to form what is becoming identified as a "global information grid". Such integration is being facilitated by the unique ability of the Standards based Web services model to support the integration of information systems irrespective of their underlying hardware, operating system or middleware structures.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 183–194, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Major security concerns arise because the system components being exposed to the Internet are often legacy systems that were not designed to operate in such a hostile environment or are, at the end-user level insecure commodity personal computer systems. Further they are being hosted within computing environments that rely on protection provided by commercially available operating systems that are known to be insecure. Commercial operating systems and their associated system tools were not designed with security in mind and the vulnerabilities that are regularly identified in such software, undermine what security they propose to provide[1]. Associated with this insecurity is a proliferation of viruses, and related malware, that take advantage of these system level vulnerabilities.

The USA's Department of Homeland Security (DHS) has emphasized the urgent need to 'build security in' in relation to software systems[2]. The homepage of their website simply states; "*Build Security In is a project of the Strategic Initiatives Branch of the National Cyber Security Division (NCSD) of the Department of Homeland Security (DHS). The Software Engineering Institute (SEI) was engaged by the NCSD to provide support in the Process and Technology focus areas of this initiative. The SEI team will develop and collect software assurance and software security information that will help software developers, architects, and security practitioners to create secure systems*"[2]. The emphasis is on security analysis and integration in the overall definition and development cycle for software and systems development.

New security paradigms, that reflect the new information security reality, are desperately needed. Given the dominance of the object-oriented programming paradigm in current application development, any attempt to provide increased the security within security aware applications (SAAs), should be applied in that context. The Self-Defending Object (SDO) philosophy defines such a new programming paradigm that aims to address part of that need; by providing a framework for the provision of mandatory access control (MAC) structures and facilities within SAAs that must operate in this new world.

This paper firstly introduces the SDO concept and then discusses the SDO Distribution Architecture (SDODA) that overcomes an major hurdle to the adoption of the SDO concept namely, its apparent incompatibility with some distributed programming models e.g. those used to develop a prototype used to support research into the SDO concept. The problem that needed to be solved was that neither the Web services nor the Java RMI models supported the passing of this special type of object to a remote procedure.

2 The SDO Concept

There is a major deficiency in the current object-oriented paradigm when applied to the development of SAAs. When an object receives a message requesting that

[1] In January 2002, Bill Gates (Microsoft) sent an email to all full time employees[1] in which he announced the company's commitment to 'trustworthy computing' and contrasted it to the current situation.

an operation be performed, the object will always attempt to satisfy that request. Whether that action should be permitted, is not considered[2]. Within the object-oriented paradigm, an object is unaware of the sensitivity of its encapsulated resources (data and/or functionality), so does not mediate requests to access those resources. Although such an approach may be appropriate for a majority of programs, it is not necessarily appropriate for SAAs. By their very nature, SAAs take security considerations into account when performing their tasks.

The SDO concept is an extension to the object-oriented paradigm that specifically targets that lack of support for the development of SAAs[3]. Specifically, that extension targets two requirements of SAAs: (1) the need to ensure that access to protected data and functionality, is in accordance with the application's security policy, and (2) all attempted accesses to those protected resources are recorded in accordance with the application's audit policy. The SDO approach aims at defining new concepts related to the growing requirement for information assurance in information systems.

An SDO is an object that encapsulates sensitive resources and has been made aware of, and assumes responsibility for its protection. Such protection is afforded through the provision of mandatory access control facilities, most likely in the form of role based access control (RBAC) structures pointing into broadly categorised permission tables or structures that implement the MAC requirements. In non-distributed SAAs, the introduction of the SDO concept has been shown to both simplify the development of SAAs and to ensure the completeness of their access control and audit measures [4].

A distinguishing characteristic of SDO methods that access protected resources, is the presence of an additional parameter, by convention the first parameter, which is an authorisation token used both as the basis of the authorisation decision (to determine whether that user is permitted to invoke that method) and to provide whatever user information is required for audit purposes.

Rather than merely acting as containers and dispensers of data and functionality, relevant software objects actively defend their sensitive resources from unauthorised access. Since in well written object-oriented programs, encapsulating objects provide the only means to access their encapsulated resources (from within the application), the enforcement of access control by those objects ensures that such measures are consistently applied and cannot be bypassed.

2.1 Limitations of the SDO Concept

The SDO concept does not, in itself, attempt to provide absolute security guarantees. It is primarily concerned with ensuring that an SAA will mediate all accesses to its protected resources, and that all such access attempts are logged.

[2] This statement is not strictly true. In Java, for example, security managers restrict access to resources that the Java runtime considers security sensitive. Mechanisms are also provided that allow access to user defined resources to be restricted. The SDO approach however, provides a framework for utilising these mechanisms to enforce the access control requirements of SAAs, in a coordinated manner.

To provide absolutely guaranteed and reliable levels of security, the major requirements that must be satisfied are: the provision of a secure computing environment (based on trusted hardware and operating system); a trusted execution environment in which application programs and any required middleware such as windowing/GUI sub-systems, network communications sub-systems and the like, will reside; and finally an application that provides complete access control.

Even in the current relatively insecure computing environments, the SDO concept has a significant role to play. By ensuring the access control measures of an SAA are systematically enforced, the rigorous application of the SDO concept will ensure that the access control requirements of the application are met. Further, by encapsulating protected resources within an SDO, rather than in a normal object, access control continues to be enforced even if the SDO is transferred between host computers. Protected information can potentially still be obtained by an attacker, if either a programming flaw exists in the SDO/ SAA code or mechanisms other than those provided by the application are employed, e.g. directly accessing the database in which sensitive information is stored.

The SDO concept, in conjunction with security mechanisms such as encryption, can provide reasonably dependable security guarantees.

Having introduced the SDO concept, the remainder of this paper will concentrate on the application of the SDO concept in distributed SAAs and the role of the SDO Distribution Architecture.

3 The HRIS Prototype

The following simplified 'Human Resources' (HR) scenario will be used to highlight aspects of the application of the SDO concept in a distributed SAA. The Human Resources Information System (HRIS) prototype stores the personnel records of a small fictitious organisation and permits transactions on that collection. The graphical user interface provides the following functionality: displaying and editing an existing personnel record, adding and deleting personnel records, and displaying all the personnel records in a tabular format.

Employees are considered to form a three level hierarchy namely; employees, managers and executives. The information kept for each employee type is:

- employee – employee number, name, work phone number, position, salary and home phone number
- manager – the above fields plus details of their supplied company car
- executive – the above fields plus details of their performance bonus.

Access to all employee information is restricted to authorised users of the HRIS. Naturally different access rules apply to different fields in an employee record and depend on the employee's position within the organisational hierarchy, e.g. access to an employee's salary is more restricted than to their phone number, and an executive's salary is more confidential than a manager's salary.

The staff of the HR department (and other authorised users) are associated with one or more of the following user RBAC roles: *hruser*, *hradmin* and

hrmanager[3]. The 'user' roles are listed in order of increasing privilege, with each role encompassing the privileges of the preceding role.

In this paper most access control decisions relate to the ability of the current user to perform a requested task so for simplicity the term 'user' will be used rather than the correct term 'principal' (that encompasses both users and processes).

3.1 HRIS Overview

As illustrated in Figure 1, HR staff normally access the information system that is located on their local area network (LAN) using the Java RMI interface, while other system users gain access across the Internet via the Web services interface[4]. Both access mechanisms provide their users with identical functionality and user interfaces. The HRIS design is based on the classic client-server architecture with each access mechanism being implemented as a separate distributed application composed of two parts, viz. a server and one or more clients. For clarity, the security-related aspects of the prototype that support authentication and authorisation have been omitted from Figure 1.

3.1.1 SDOs and Other Important HRIS Objects

The more significant SDO and object classes used within the HRIS are:

- the employee objects that encapsulate individual personnel records, a hierarchy of the classes *Employee*, *Manager* and *Executive*;
- the datastore objects that provide restricted access to the personnel database, class *DataStore*;
- the RMI server object that provides restricted access to 'the server' via the RMI interface, class *EmployeeServerImpl*;
- the Web services server that provides restricted access to 'the server' via the Web services interface, class *EmployeeWebServerImpl*;
- the class used to convert between the SDOs and their non-SDO equivalents, class *SDOClassConverter*[5];
- the employee list object that encapsulates a list of personnel records, class *StringArrayList*; and
- The authorisation credential that encapsulates the user's unique identifier and current role, (non SDO) class *UserID*.

[3] There are additional administrative roles associated with system administrators or for internal use by the system.

[4] Two modes of access to the HRIS were investigated to ensure that any success in applying the SDO approach was not dependent on the programming model chosen. The RMI interface was chosen to represent the traditional, proprietary-based distributed computing approach because it is commonly used and fully integrated into the Java API. The Web services model was chosen as it is becoming the distributed programming model of choice and is supported by the majority hardware vendors and software tool developers.

[5] The need to perform such class conversions is identified in Section 5.1.1.

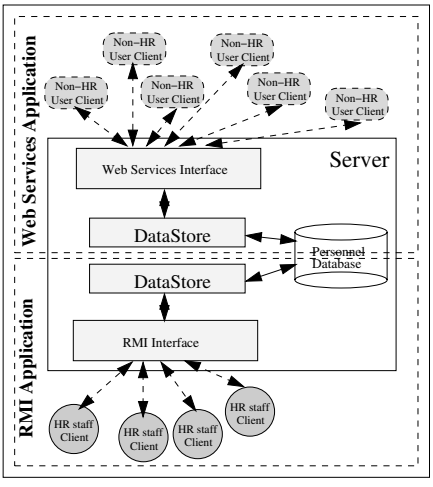


Fig. 1. HRIS Overview

To achieve separation of application and authorisation logic, an SDO will typically delegate responsibility for making authorisation decisions to a separate authorisation object (AO). In the HRIS, each SDO encapsulates a dedicated AO. That model was chosen because it was believed to be the most difficult to implement. When using this model, whenever an SDO was transferred between the client and server both the SDO and its encapsulated AO need to be transferred. Thus any problems that might prevent the use of the SDO concept in a distributed environment should be identified. Alternative approaches involve some degree of sharing of AO instances e.g. all SDOs belonging to the same inheritance hierarchy share the same AO. If such models were used, only the SDO would need to be transferred between machines. Upon arrival at its destination, the SDO would be associated with the relevant shared AO at its destination.

4 The Web Services Programming Model

The Web services model is becoming popular as it is Standards based and employs XML-based messaging that provides implementation independence. Like other Web services toolkits, the Java Web Services Developer Pack (JWSDP) that was used to develop the HRIS, was created to enable the easy integration of a Web service into a distributed application. From the WSDL description of the service, all the classes needed to support remote procedure call semantics, are automatically generated. The details of the remote method call being made (including the identification of the method being invoked and its parameters) are encoded in XML within a SOAP message.

The Simple Object Access Protocol (SOAP) is a standards-based protocol for invoking Web services through the exchange of XML messages, i.e. enables

remote procedure calls over the Internet. Roy[5] summarised the SOAP specification as: "*The SOAP specification defines an envelope for transmitting messages, offers guidelines for encoding data, and provides rules for representing remote-procedure calls*". The specification of standards, including WS-Security [6, 7] has resulted in toolkit support for the securing SOAP messages at both the transport and application layers.

The JWSDP toolkit does not employ any system level mechanisms (such as serialization which is employed by the Java RMI). Instead when an object is passed as a parameter to a remote method, the SOAP message is constructed by 'reading' the value of each of the object's data members using the relevant accessor method. Similarly, at its destination, the object is reconstructed by instantiating a new object and then initialising it by invoking each of its mutator methods in turn (using data from the SOAP message).

4.1 The Apparent Incompatibility of SDOs and Web Services Tools

As specified in the JWSDP documentation, any object which is to be passed as a parameter to, or is returned by, a remote call must have the following properties:

1. the class must provide an appropriately named accessor and mutator method for every data member, and
2. the class must provide a default (i.e. zero parameter) constructor.

These requirements are met by most classes that are likely to be passed to remote methods so the remote method invocation requirements of non-SDO based applications are supported by the JWSDP. However SDOs cannot be transferred. The Web services infrastructure uses accessor and mutator calls of the object being transferred to disassemble and later 'reconstruct' the object at its destination. The accessor and mutator methods of SDOs perform authorisation and have a different signature, so do not conform to the JWSDP requirements. In fact the design of SDOs is orthogonal to such requirements as the defining characteristic of an SDO is that it performs access control on its sensitive resources. The presence of SDO accessor and mutator methods that access protected data without performing authorisation, would circumvent the SDO's protection mechanism.

4.2 The SDO Distribution Architecture (SDODA) Provides a Solution

The introduction of the SDO Distribution Architecture (SDODA) overcomes this incompatibility between the Web services model and the use of SDOs. The problem is that the Web services infrastructure is unable to use accessor and mutator methods of SDOs, so the solution is to raise the level of abstraction provided to the application, hence the SDO-based application interacts with the SDODA's middleware rather than directly with the Web services infrastructure. The SDODA's trusted middleware creates a 'nonSDO' object[6] from any SDO

[6] The term 'nonSDO' is used to denote an object that only provides constructors, standard accessor and mutator methods, and encapsulates exactly the same information as its corresponding SDO.

that needs to be passed between the Web services client and server. The nonSDO parameters, along with the other (unmodified) parameters, are then passed to the corresponding Web services infrastructure method. At its destination, an SDO is recreated from the transferred nonSDO by the middleware. Thus, the SDODA enables SDOs, as well as normal objects and primitive types, to be used as parameters or the return value of remote method invocations.

It might appear that the proposed architecture will cause a loss in security because at both the client and server, the security sensitive content is now encapsulated within nonSDOs (as well as SDOs). However, even though nonSDOs would divulge their sensitive content should one of their accessor methods be invoked, access to their references is controlled. All references to nonSDOs are kept private within the middleware layer (except for being passed to the necessarily trusted Java runtime) so any nonSDOs become non-reachable as soon as the middleware call completes. At the server, the situation is similar. Hence the possibility that an attacker will obtain a reference to a nonSDO is minimal. Thus the SDOSA increases the level of abstraction available to the application thus enabling the Web services infrastructure to support the passing of SDOs between the client and server components of the application while ensuring that the SDO's content continues to be protected from unauthorised access.

In the next section, the realisation of the proposed architecture in a Web services environment will be discussed. Although not discussed in this paper, SDOs also experience a non-related incompatibility problem with Java RMI, which is also overcome using the SDODA mechanisms.

5 Implementing the SDO Distribution Architecture

The essence of the architecture is the introduction of a middleware layer that sits between the distributed programming model infrastructure and the SDO-based distributed SAA and mirrors the services provided by the underlying infrastructure. Those methods only differ in the types of their parameters with the middleware accepting SDOs while the underlying infrastructure only accepts the equivalent nonSDO.

Whenever a SDO needs to be passed as a parameter to a remotely invoked method (or returned from such an invocation), the SDO-based SAA invokes the relevant method of the middleware potentially passing one or more SDOs as parameters. That method creates the corresponding nonSDO object from each SDO parameter and passes them, along with the other parameters, to the corresponding server method of the programming model infrastructure.

At the destination, the relevant middleware method is invoked by the underlying programming model infrastructure. The middleware performs the necessary nonSDO to SDO parameter conversion, and then invokes the SDO-based server method. Should an SDO need to be returned, the middleware is responsible for enabling the return of that value by effectively the reverse process.

5.1 The Web Services Version of the HRIS Prototype

An overview of the Web services version of the HRIS that shows the implementation of the SDODA, is presented in Figure 2.

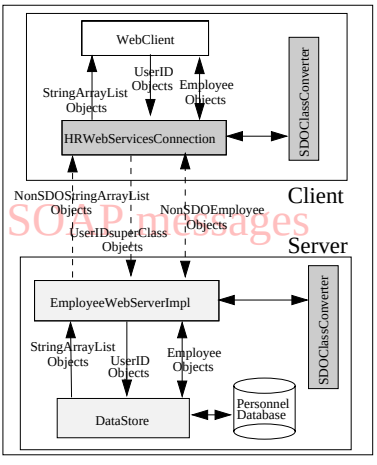


Fig. 2. Transferring SDOs in a Web Services Environment

5.1.1 The Client Solution

On the client side, the middleware is implemented as the *HRWebServicesConnection* class. To access a remote service, the client application invokes a *HRWebServicesConnection* method, rather than a method of the 'real' server. Developers of the client application are able to pass SDOs as parameters to those methods and/or receive SDOs as return values, e.g, the method *get* returns the requested *Employee* SDO.

The *HRWebServicesConnection* methods mirror those provided by the server, differing only in the type of the parameters being passed and return values when an SDO is involved. The methods of the middleware (class *HRWebServicesConnection*) exhibit common behaviour that can be summarised as follows: for each SDO passed as a parameter, instantiate an instance of the corresponding nonSDO using the information content of the SDO, then invoke the corresponding server method. Normal objects and primitive type parameters are forwarded unchanged. Return values are handled by reversing the process. In the prototype, the task of performing the required type conversions between SDOs and nonSDOs is delegated to a separate *SDOClassConverter* object.

The methods of the *SDOClassConverter* class need to invoke the constructors together with the accessor and mutator methods of all of the SDO classes that they convert. Consequently, that middleware object requires sufficient privileges to invoke those SDO methods. In the prototype, those privileges are associated with an administrative role *SU* and the administrator who executes the client program (and similarly the server program) is required to authenticate into that

role. The authorisation token generated is passed to and used exclusively by the middleware software[7]. To execute a client (in the prototype at least), the administrator must authenticate into the role *creator* (to acquire the privileges required to instantiate the SDOs needed to initialise the client side application) and then authenticate into the role *SU* (to provide the credential needed by the system). Then, the administrator or another user must authenticate into the relevant user role to gain the privileges needed to use the application.

In the HRIS, both the *HRWebServicesConnection* and the *SDOClassConverter* classes were 'hand coded'. In an SDO production environment, the SDO Support Environment would be expected to include a software tool that would automatically generate these classes and their server-side equivalents.

5.1.2 The Server Solution

On the server side, a analogous approach is used to that employed on the client side. In HRIS, the class *EmployeeWebServerImpl* provides both an implementation of the server interface is responsible for interfacing the non-SDO aware Web services infrastructure with the SDO-aware server application. *EmployeeWebServerImpl* is an SDO as it provides protected functionality, namely the ability to invoke server methods[8]. Once the required parameter translations are performed, the appropriate *DataStoreImpl* method is invoked to perform the requested operation on the personnel database (after performing another authorisation check as it too is an SDO). As shown in Figure 2, the conversion of parameters at the server is delegated to an instance of the same *SDOClassConverter* class used by the client.

5.2 Special Classes for the Web Services Prototype

The applications running on both the client and server machines are based on the SDO paradigm, i.e. they know about and use SDOs of types *Employee* and *StringArrayList*, and also use *UserID* objects (see Section 3.1.1). Since the Web services infrastructure is unable to handle SDOs, objects of the corresponding WSDP compliant classes; *NonSDOEmployee*, *NonSDOArrayList* and *UserIDsuperClass*[9] are passed between the server and its clients. The SDODA middleware provides the interconnection of these otherwise incompatible subsystems.

5.2.1 SDO and 'NonSDO' Classes

Even though SDOs and their nonSDO equivalents are both effectively holders for the same information, an SDO such as a *Manager* object, is fundamentally different to its non-SDO equivalent, a *NonSDOEmployee* object. A *Manager*

[7] The alternative approach would involve the system generating the required authorisation credential using a mechanism that bypasses the authentication subsystem.

[8] It is assumed that the resources expended in performing the authorisation check is significantly less than those expended by the server in processing unauthorised requests.

[9] Those WSDP compliant classes are automatically generated from the WSDL description of the Web service which includes a description of those classes.

SDO enforces access control on its encapsulated protected content by restricting access to sensitive methods. It is also a member of a class hierarchy and it attempts to monitor the integrity of its protected content[10]. In contrast, a *NonSDOEmployee* object only provides constructors and accessor and mutator methods for each of its data members.

The small linear SDO class hierarchy used in the HRIS allowed the use of a single *NonSDOEmployee* class to act as the nonSDO equivalent for all of the *Employee* classes. That class encapsulates all of the data members that are declared anywhere in the *Employee* hierarchy plus an additional data member which specifies the type of the *Employee* object that it represents. A static method *SDOClassConverter.convertToSDOEmployee* is used to convert a *NonSDOEmployee* object to the appropriate type of *Employee* object. That conversion involves obtaining the value of each data member from the nonSDO object and then using those values to instantiate the *Employee* object. The other class conversions are performed in a similar manner. The systematic nature of the method's code supports the previously expressed view that a software tool could be developed to generate the class (or set of classes) needed to perform the type conversions required to support the operation of a particular server.

By introducing a level of abstraction above that provided by the standard Web services infrastructure, the middleware software facilitates the transfer of SDOs between the individual systems on which a distributed SDO-based SAA resides.

5.2.2 Authorisation Credentials

Authorisation credentials (*UserID* objects) are also passed as parameters to the remote methods, and also require a Web services compliant counterpart, namely *UserIDsuperClass* objects. Both classes contain identical information namely: the *role* of the user, the *userName* of the user (a unique system wide identifier for a user that is needed for audit purposes), the *expiryDate* of the token, and a *signature* generated by the authentication service.

In this case, the problem is not that the *UserID* is an SDO, but rather that *UserID* objects provide additional functionality. Not being an SDO, the class *UserID* can extend the *UserIDsuperClass* class to provide the required integrity checking. This approach is needed as the class *UserIDsuperClass* is automatically generated so cannot perform custom functionality.

6 Conclusions

The applicability of the SDO concept to the development of non-distributed SAAs has been previously demonstrated[4]. Consequently the SDO concept can also be applied to the development of the applications that run on the client and server machines. However to be truly applicable to the development of distributed SAAs, the passing of SDOs as parameters to remote procedure calls

[10] The use of cryptography to provide integrity guarantees within the *Employee* and *UserID* classes is not considered in this paper.

must also be supported. The proposed SDODA overcomes the apparent incompatibilities between SDOs and the Web services and Java RMI distributed programming models.

As described in Section 5.1, the development of the Web services version of the HRIS prototype demonstrated that SDODA could be used to support distributed SAAs that employ the Web services model. The provision of trusted middleware that converts between SDOs and normal objects, allowed the remote method invocation support provided by Java's JWSDP to be harnessed, to allow SDOs to be passed to and from a Web services server. A second HRIS application based on the Java RMI model, has also been successfully developed using the proposed architecture.

It is believed that since the trusted middleware is sandwiched between the Web services (or Java RMI) infrastructure layer and the SDO-based application, and essentially provides a parameter translation service, the proposed architecture should support distributed SDO-based SAAs based on other distributed programming models. Further, since the implementation of the distributed SDO-based prototype is not believed to be dependent on any Java specific language constructs or API support, the SDO concept and the SDODA should be more generally applicable. Further research is needed to validate our belief that the SDO concept is applicable to the development of distributed SAAs using other platforms such as .Net, and other distributed paradigms such as CORBA.

References

1. Gates, B.: Bill Gates: Trustworthy Computing. Wired News (2002) `http://www.wired.com/news/print/0,1294,49826,00.html`.
2. Department of Homeland Security: 'Build Security In' home page. (2006) http://buildsecurityin.us-cert.gov.
3. Holford, J.W., Caelli, W.J., Rhodes, A.W.: The concept of self-defending objects in the development of security aware applications. In: 4th Australian Information Warfare and IT Security Conference, Adelaide, Australia (2003)
4. Holford, J.W., Caelli, W.J., Rhodes, A.W.: Using self-defending objects to develop security aware applications in Java. In Estivill-Castro, V., ed.: 27th Australasian Computer Science Conference. Volume 26 of Conferences in Research and Practice in Information Technology., Dunedin, New Zealand, Australian Computer Society (2004) 341–349
5. Roy, J., Ramanujan, A.: Understanding web services. IT Pro (2001) 69–73
6. IBM, Microsoft: Security in a web services world: A proposed architecture and roadmap, version 1.0. Technical report, IBM and Microsoft (2002)
7. Nadalin, A., Kaler, C., Hallam-Baker, P., Monszillo, R.: Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). Technical report, OASIS, http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0 (2004)

Efficient and Provably Secure Multi-receiver Identity-Based Signcryption*

Shanshan Duan and Zhenfu Cao

TDT Laboratory, Shanghai Jiao Tong University,
200240, Shanghai, P.R. China
dss@sjtu.edu.cn, zfcao@cs.sjtu.edu.cn

Abstract. In this paper, we propose an efficient multi-receiver identity based signcryption scheme which only needs one pairing computation to signcrypt a message for n receivers and can provide confidentiality and authenticity simultaneously in the multi-receiver setting. We compare our scheme with several multi-receiver constructions from the security and efficiency points of view and argue that our provably secure scheme is more efficient than other known constructions. To address the security issues, we formulate security models and define strong security notions for multi-receiver identity based signcryption schemes. We also prove that our scheme satisfies these strong security requirements in the random oracle model.

1 Introduction

The multi-receiver setting for public key encryption is that there are n receivers, numbered $1, ..., n$, and each of them generates for itself a private and public key pair denoted by (sk_i, pk_i). A sender encrypts a message M_i using pk_i to obtain C_i for $i = 1, ..., n$ and then sends $(C_1, ..., C_n)$ as a ciphertext. Upon receiving the ciphertext, receiver i extracts C_i and decrypts it using isk_i. The concept of this multi-receiver setting was formalized by Bellare et al. [1]. They proved that in the sense of indistinguishability, public key encryption in the multi-receiver setting is secure if it is secure in the single-receiver setting. Baudron et al. [2] also proved the same result independently. To save the bandwidth and minimize the computation cost, Kurosawa [3] showed one could design multi-receiver encryption schemes using a technique called "randomness re-use". Bellare [4] broadened Kuraosawa's investigation. He specified an appropriate security model and provided a test to determine whether a public key encryption scheme permits secure randomness re-use to build up a multi-receiver encryption scheme.

Identity based cryptosystem was introduced by Shamir in 1984 [5]. Its main idea is that public keys can be derived from arbitrary strings while private keys

* This work is supported in part by the National Natural Science Foundation of China for Distinguished Young Scholars under Grant No. 60225007 and 60572155, the Science and Technology Research Project of Shanghai under Grant Nos. 04JC14055 and 04DZ07067, and the Special Research Funds of Huawei.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 195–206, 2006.
© Springer-Verlag Berlin Heidelberg 2006

can be generated by the trusted Private Key Generator(PKG). In such systems, there is no need to bind a public key to its owner's identity. So trust problems encountered in the certificate based public key infrastructures do not exist. Baek et al. [6] incorporated identity based encryption to the above multi-receiver setting and called the interesting result "multi-receiver identity based encryption".

In this multi-receiver identity based setting, we are interested in the situation where there are not only multiple receivers but also multiple senders. As an example, consider that there are several managers, each of whom wants to securely broadcast an e-mail to the employees of the company independently. Once an employee receives several ciphertexts from different managers, an issue of message authentication will arise. In such cases, it is desirable to achieve confidentiality and authenticity simultaneously, so each manager's e-mail should be encrypted and authenticated. One might argue that by adding sender authentication to Baek's scheme these security requirements can be satisfied at the same time. However, such combinations may suffer from hidden security weakness and we will discuss them in Section 6. Recall that in the single-receiver setting, signcryption [7], whose idea is to perform signature and encryption in a single logical step, is an efficient way to obtain confidentiality, authenticity, integrity and non-repudiation. Therefore in the multi-receiver identity based setting, we might also introduce signcryption to achieve these security goals.

In the course of constructing multi-receiver identity based signcryption scheme, special attention must be paid to efficiency**.** So far, there are two generic methods to build multi-receive schemes, one is to simply process a message for each receiver, and the other is to use the technique of randomness re-use [3]. Considering that almost all the ID-based signcryption schemes are based on bilinear pairings, no matter which method is used, n bilinear pairing computations are required to signcrypt a message for n receivers. This is far from efficient. Besides, these two methods are both applied to the standard single-receiver schemes and a scheme's security in the single-receiver setting is different from that in the multi-receiver setting. So given a single-receiver scheme, we can not make sure whether it admits secure use of the above two methods. To meaningfully address the security issues of MIBSC schemes, we need appropriate models and definitions of security.

Our contributions. Following the above discussion, a natural question is how to design a multi-receiver identity-based signcryption(MIBSC) scheme with a high-level of computational efficiency while achieving authenticity and confidentiality simultaneously. In this paper, we answer this question by making the following contributions:

1. We present an efficient multi-receiver IBSC scheme that only requires *one* pairing computation to signcrypt a single message for multiple receivers.

2. We specify a security model based on the *selective multi-identity attack* model in which the adversary commits ahead of time to multiple identities which it intends to attack and formalize two security notions for MIBSC schemes. The notions are confidentiality against chosen ciphertext attacks and strong

existential unforgeability against chosen message attacks respectively. The latter also supports ciphertext authentication so it can be viewed as a strengthening of Boyen's signature non-repudiation [8].

3. We prove that our scheme satisfies the above security notions in the random oracle model [9]. To be specific, its chosen ciphertext security is shown to be related to the hardness of BDH problem while its unforgeability is related to the hardness of CDH problem.

4. We present certain possible MIBSC constructions and compare our scheme with them from security and efficiency points of view respectively. We conclude that our provably secure scheme has better performance.

2 Preliminary

2.1 Pairings

We briefly review the necessary facts about bilinear maps. Let us consider groups G_1 and G_2 of the same prime order q, writing the group action multiplicatively. A bilinear map is a map $e : G_1 \times G_1 \rightarrow G_2$ satisfying the following properties:

1. Bilinearity: $\forall P, Q \in G_1$, $\forall a, b \in Z_q$, $e(P^a, Q^b) = e(P, Q)^{ab}$.
2. Non-degeneracy: for any point $P \in G_1$, $e(P, Q) = 1$ for all $Q \in G_1$ iff $P = \mathcal{O}$.
3. Computability: $\forall P, Q \in G_1$, $e(P, Q)$ can be efficiently computed.

Now recall some candidate hard problems that provide underlying assumptions from pairings that will be used later.

Definition 1. *(Bilinear Diffie-Hellman(BDH) problem)The BDH problem is, given P, P^a, P^b, $P^c \in G_1$, for unknown $a, b, c \in Z_q^*$, to compute $e(P, Q)^{abc}$.*

The advantage of any probabilistic polynomial time(PPT) algorithm $\mathcal{G}$ in solving BDH problem in G_1 is defined to be:

$Adv_{\mathcal{G}}^{BDH} = \Pr[\mathcal{G}(P, P^a, P^b, P^c) = P^{abc} : a, b, c \in Z_q^*]$.

BDH assumption: For every PPT algorithm $\mathcal{G}$, $Adv_{\mathcal{G}}^{BDH}$ is negligible.

Definition 2. *(Computational Diffie-Hellman(CDH) problem)The CDH problem is, given P, P^a, $P^b \in G_1$ for unknown $a, b \in Z_q^*$, to compute P^{ab}.*

The advantage of any probabilistic polynomial time(PPT) algorithm $\mathcal{G}$ in solving CDH problem in G_1 is defined to be:

$Adv_{\mathcal{G},G_1}^{CDH} = \Pr[\mathcal{G}(P, P^a, P^b) = P^{ab} : a, b \in Z_q^*]$.

CDH assumption: For every PPT algorithm $\mathcal{G}$, $Adv_{\mathcal{G},G_1}^{CDH}$ is negligible.

2.2 Identity-Based Signcryption

Malone-Lee [10] proposed the first ID-based signcryption scheme along with a security model. This model dealt with notions of privacy and unforgeability. However, his scheme is not semantically secure. To overcome this weakness, Quisquater [11] proposed a new ID-based signcryption scheme that achieves

both public verifiability and semantic security against chosen ciphertext attacks. Boyen [8] developed the security model of [10] by adding several new security notions. Recently, Chen and Malone-Lee [12] present an ID-based signcrpytion which claimed to be the most efficient, provably-secure scheme of its type proposed to date.

Generally, an ID-based signcryption scheme consists of four basic algorithms: Setup, Extract, Signcrypt, De-signcrypt. Setup generates common public parameters and the master key; Extract generates the private key for each user according to its identity; Signcrypt produces a ciphertext from a sender to a designated receiver; De-signcrypt checks the integrity of received ciphertext and recovers the original message.

3 Definitions of Multi-receiver Identity-Based Signcryption

3.1 Syntax

In the setting of multi-receiver identity based signcryption, either a single message or multiple messages can be signcrypted. In our context, we assume that a single message is signcrypted to multiple receivers.

Definition 3. *A generic multi-receiver identity based signcryption(MIBSC) scheme consists of the following algorithms.*

Setup: Given a security parameter, the private key generator(PKG) generates a master key mk_{PKG} and a common parameter cp_{PKG}. cp_{PKG} is given to all interested parties while mk_{PKG} is kept secret.

Extract(private key extraction): Providing an identity ID received from a user and its master key mk_{PKG} as input, the PKG runs this algorithm to generate a private key associated with ID, denoted by S_{ID}.

Signcrypt: To send a message m to multiple receivers whose identities are ID_1, ..., ID_n respectively, the sender runs this algorithm to generates a signcrypted ciphertext $C = Signcrypt(M, S_{ID_S}, ID_1, ..., ID_n)$.

De-signcrypt: Upon receiving a ciphertext C, the receiver ID_i computes $De-signcrypt(C, S_{ID_i}, ID_S)$ and obtains $m \in M \cup \{\perp\}$, where $\perp$ indicates that the message was not encrypted or signed properly.

In our approach, a multi-receiver signcrypted ciphertext is a combination of two parts. The first part is same to all the receivers and the second part can be viewed as an n-tuple where the i-th component is specific to receiver ID_i. When decrypting a ciphertext, receiver ID_i extracts the first part and the i-th component from the second part and then runs the De-signcryption algorithm. We refer to the former as receiver information part and the latter as text part. We adopt this approach because it simplifies the security model and allows for a clear explanation of the security notions.

3.2 Security Notions

Now we present security notions for multi-receiver IBSC schemes. Recall that the selective identity attack, which was first proposed by Canetti et al. [13], means that an adversary commits ahead of time to the identity on which it will be challenged. We extend this notion to the multi-receiver setting and refer to it as *selective multi-identity attack*. Thus in our models, the adversary is assumed to output ahead of time multiple identities that it wishes to attack. Besides, due to the identity based nature, we should assume that the adversary may obtain any private key other than those of the multiple target identities. Note that in the description of the models, we often equate a user with its identity.

Message Confidentiality. With respect to confidentiality, the widely accepted notion is indistinguishability of ciphertexts under chosen ciphertext attacks. We adapt it to the multi-receiver setting and refer to it as *indistinguishability of ciphertexts under selective multi-ID, chosen ciphertext attack(IND-sMIBSC-CCA)*. Generally, a ciphertext in the multi-receiver setting can be viewed as a combination of text part and receiver information part. In the game defined for confidentiality, if an adversary simply modifies some components in the receiver information part of the challenge ciphertext and submits the changed ciphertext to the de-signcryption oracle on an identity whose corresponding component in the receiver information part is not modified, then it can obtain the message and win the game. The intuition is that the original challenge and the modified challenge look the same to the queried identity. To avoid this trivial attack, we do not allow the adversary to issue such de-signcryption queries after the challenge phase of the game. The notion of *IND-sMIBSC-CCA*, more restrictive but still reasonable, is defined as follows:

Definition 4. *We say that a multi-receiver ID-based signcryption scheme is secure against chosen ciphertext attacks(IND-sMIBSC-CCA) if no probabilistic polynomial time adversary has a non-negligible advantage in the following game:*

Setup: The challenger $\mathcal{B}$ runs the Setup algorithm to generate a master key and a common parameter (mk_{PKG}, cp_{PKG}). $\mathcal{B}$ gives cp_{PKG} to $\mathcal{A}$ while he keeps mk_{PKG} secret from $\mathcal{A}$. After receiving the system parameters, the adversary $\mathcal{A}$ outputs multiple target identities, denoted by $ID_1^*, ..., ID_n^*$ respectively.

Phase 1: $\mathcal{A}$ issues a first series of queries of the following kinds adaptively:

- Private key extraction queries: $\mathcal{A}$ produces an identity ID and requires its private key, the challenger runs the private key extraction algorithm to get $S_{ID} = Extract(mk_{PKG}, ID)$. A restriction here is that $ID \neq ID_i^*$ for $i = 1, ..., n$.
- Signcryption queries: $\mathcal{A}$ produces a message $m \in \mathcal{M}$, n receivers' identity $ID_{R_1}, ..., ID_{R_n}$ and requires the result of $Signcrypt(m, ID_j^*, ID_{R_1}, ..., ID_{R_n})$ for some attacked user's private key $S_{ID_j^*} (j \in [1, n])$.

- De-signcryption queries: $\mathcal{A}$ produces a ciphertext σ and requires the result of $De - signcrypt(\sigma, S_{ID_i^*})$ for some $i \in [1, n]$. This result is made of a signed plaintext and a sender's public key if the obtained signed plaintext is valid for the recovered sender's public key. Otherwise, the $\perp$ symbol is returned as result(indicating that the ciphertext was not properly formed).

Challenge: $\mathcal{A}$ produces two equal length plaintexts m_0, $m_1 \in \mathcal{M}$ and an arbitrary private key S_{ID_S}. $\mathcal{B}$ flips a coin $b \leftarrow \{0, 1\}$ to compute a signcryption $\sigma = Signcrypt(m_b, S_{ID_S}, ID_1^*, ..., ID_n^*)$ with the sender's private key S_{ID_S} under the attacked public keys $ID_1^*, ..., ID_n^*$. σ is sent to $\mathcal{A}$ as a challenge.

Phase 2: $\mathcal{A}$ issues new queries as in Phase 1. It can not ask the de-signcryption of the challenge σ with the private key $S_{ID_i^*}$ for any $i \in [1, n]$ nor query the de-signcryption oracle on an identity and a ciphertext σ' which is only different from σ in the receiver information part.

Guess: At the end of the game, $\mathcal{A}$ outputs a bit b' and wins if $b' = b$.

$\mathcal{A}'s$ advantage is defined to be $Adv^{IND-sMIBSC-CCA}(\mathcal{A}) := 2\Pr[b' = b] - 1$.

Strong Existential Unforgeability. Usually, the second notion for signcryption is unforgeability which can only provide signature non-repudiation. In our security model, we add support for authenticated encryption and call the resulting notion strong existential unforgeability. This security requirement makes sure that the sender can not deny having signcrypted a message to the receivers and so the sender and the encryptor are guaranteed to be the same person. Considering that in our context an adversary is a selective multi-identity one, we allow it to forge a ciphertext on behalf of any of the multiple target identities while knowing all the multiple receivers' private keys. This notion, called *strong existential unforgeability under selective multi-ID, chosen message attack(SUF-sMIBSC-CMA)*, is defined as follows.

Definition 5. *We say that a multi-receiver ID-based signcryption scheme is said to be strongly existentially unforgeable against chosen-message attacks (SUF-sMIBSC-CMA) if no PPT forger has a non-negligible advantage against a challenger $\mathcal{B}$ in the following game:*

Setup: The challenger $\mathcal{B}$ runs the Setup algorithm to generate a master key and a common parameter (mk_{PKG}, cp_{PKG}). $\mathcal{B}$ gives cp_{PKG} to $\mathcal{F}$ while he keeps mk_{PKG} secret from the forger $\mathcal{F}$. After receiving the system parameters, $\mathcal{F}$ outputs multiple target identities, denoted by $ID_1^*, ..., ID_n^*$ respectively.

Attack: $\mathcal{F}$ issues queries to the same oracles as those in Definition 4.

Forgery: $\mathcal{F}$ eventually produces a ciphertext σ and n arbitrary receivers' key pairs $(ID_{R_1}, S_{ID_{R_1}})$,...,$(ID_{R_n}, S_{ID_{R_n}})$. $\mathcal{F}$ wins if σ decrypts under any private key of these receivers, to be a signed message (m, s, ID_j^*) for some $j \in [1, n]$ that satisfies the pair (m, s) is valid for the public key ID_j^* and σ was not the output of a signcryption query $Signcrypt(m, ID_j^*, ID_{R_1}, ..., ID_{R_n})$.

4 Our Construction

In this section, we present our construction from pairings.

Setup: On inputting a security parameter $k \in N$, the PKG chooses the system parameters that include two groups G_1, G_2 of prime order $q \geq 2^k$, a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, a generator $P \in G_1$ and a random $s \in Z_q$ as a master key. Then it sets $P_{TA} = P^s \in G_1$ as the system public key. The PKG also chooses cryptographic hash functions $H_0 : \{0,1\}^t \rightarrow G_1$, $H_1 : \{0,1\}^t \times G_1 \times \{0,1\}^t \rightarrow Z_q$, $H_2 : G_2 \rightarrow \{0,1\}^{2t}$, $H_3 : G_2 \times \{0,1\}^t \rightarrow G_1$. The system parameters are $\langle q, G_1, G_2, e, P, P_{TA}, H_0, H_1, H_2, H_3 \rangle$.

Extract: Given a user identity string $ID \in \{0,1\}^*$, his public key is $Q_{ID} = H_0(ID)$. His private key is $S_{ID} = (Q_{ID})^s$ which is calculated by the PKG.

Signcrypt: Suppose Alice whose identity is ID_A wants to signcrypt a message m to n different receivers $ID_1, ..., ID_n$.

- Sign:
  Alice chooses $r \in Z_q^*$, $Q_R \in G_1$ randomly and computes:
  $X = P^r$, $h = H_1(m, X, Q_R)$, $Q_A = H_0(ID_A)$ and $W = S_A^h Q_A^r$ where $S_A = Q_A^s$ is Alice's private key.
- Encrypt:
  Then Alice computes $V = e(P_{TA}^r, Q_R)$, $Y = H_3(V, ID_A) \cdot W$, $Z = H_2(V) \oplus (ID_A \parallel m)$, $U_i = (Q_R Q_{ID_i})^r (i = 1, \ldots, n)$ where $Q_{ID_i} = H_0(ID_i)$.

The ciphertext is $\sigma = (X, Y, Z, Q_R, U_1, ..., U_n)$.

De-signcrypt: Each receiver ID_i uses his private key to decrypts σ.

- Decrypt:
  Assume the private key of ID_i is S_{ID_i}. ID_i computes:
  $V' = e(P_{TA}, U_i) \cdot e(X, S_{ID_i})^{-1}$, $(ID_A \parallel m) = H_2(V') \oplus Z$.
  Then output (ID_A, m) together with $\langle X, Y, V', Q_R \rangle$ to verify.
- Verify:
  ID_i computes $W' = Y \cdot H_3(V', ID_A)^{-1}$.
  Compare if: $e(P, W') = e(X P_{TA}^h, Q_A)$ where $h = H_1(m, X, Q_R)$ and $Q_A = H_0(ID_A)$.

Output m if the above verification is true, or output $\perp$ if false.

Correctness: It is easy to see that the above De-signcrypt algorithm is consistent. If σ is a valid ciphertext, since

$$\begin{aligned} V' &= e(P_{TA}, U_i) \cdot e(X, S_{ID_i})^{-1} \\ &= e(P_{TA}, (Q_R Q_{ID_i})^r) \cdot e(X, S_{ID_i})^{-1} \\ &= e(P_{TA}, Q_R^r) \cdot e(P_{TA}, Q_{ID_i}^r) \cdot e(P^r, Q_{ID_i}^s)^{-1} \\ &= e(P_{TA}^r, Q_R), \end{aligned}$$

then receiver ID_i can decrypt the ciphertext and obtain the signed message.

From the description of algorithm De-signcrypt we know that for receiver ID_i, it only needs to extract the i-th component U_i from $(U_1, ..., U_n)$ and then uses U_i and its private key to obtain the sender's signature from (X, Y, Z, Q_R). So for a ciphertext $(X, Y, Z, Q_R, U_1, ..., U_n)$, (X, Y, Z, Q_R) is the text part while $(U_1, ..., U_n)$ is the receiver information part.

5 Proof of Security

Theorem 1. *In the random oracle, if an adversary $\mathcal{A}$ has non-negligible advantage ϵ against the IND-sMIBSC-CCA security of our scheme when running in time t and performing q_{SC} signcryption queries, q_{DSC} de-signcryption queries and q_{H_i} queries to oracles H_i(for $i = 0, 1, 2, 3$), then there is an algorithm $\mathcal{B}$ that solves the BDH problem with probability $\epsilon' \geq \frac{1}{q_{H_2}}\left(\epsilon - q_{H_2} q_{DSC}/2^{2k}\right)$ and within running time $t' < t + (2q_{DSC} + q_{SC})\, t_e$ where t_e denotes the time required for one pairing evaluation.*

Proof. We show how to build an algorithm $\mathcal{B}$ that solves the BDH problem by running the adversary $\mathcal{A}$ as a subroutine. On input $(P, P^{\alpha}, P^{\beta}, Q = P^{\gamma})$, $\mathcal{B}$'s goal is to compute $e(P,P)^{\alpha\beta\gamma} = e(P,Q)^{\alpha\beta}$. W.l.o.g., we assume that for any ID, $\mathcal{A}$ queries H_0 and private key extraction oracle at most once, and $\mathcal{A}$ queries $H_0(ID)$ before ID is used as an input of any other queries. To handle $\mathcal{A}$'s queries, $\mathcal{B}$ maintains lists L_i to keep track of the answers given to oracle queries on $H_i (i = 0, 1, 2, 3)$. $\mathcal{B}$ plays the role of $\mathcal{A}$'s challenger and works by interacting with $\mathcal{A}$ in a game defined as follows:

Setup: $\mathcal{B}$ sends the system parameter to $\mathcal{A}$ with $P_{TA} = P^{\beta}$. Then $\mathcal{A}$ outputs multiple target identities, denoted by $(ID_1^*, ..., ID_n^*)$.

Phase 1: $\mathcal{A}$ performs a first series of queries of the following kinds that are handled by $\mathcal{B}$ as explained below:

[Query on H_0 for identity ID_j]: If there exists (ID_j, λ_j, E_j) in L_0 list, return E_j. Otherwise, do the following:

- If $ID_j = ID_i^*$ for some $i \in [1, n]$, choose $\lambda_i^* \in Z_q^*$ uniformly at random and compute $E_j = P^{\lambda_i^*} Q^{-1}$, else choose $\lambda_j \in Z_q^*$ uniformly at random and compute $E_j = P^{\lambda_j}$.
- Put (ID_j, λ_j, E_j) into L_0 and return E_j as the answer.

[Queries on $H_i (i = 1, 2, 3)$]: Produce a random element from the appropriate range, and add both query and answer to the corresponding list.

[Private key extraction query on ID_j]: Note that in our defined model $ID_j \neq ID_i^*$ for some $i \in [1, n]$. B recovers (ID_j, λ_j, E_j) from the list L_0, compute $S_{ID_j} = (P^{\beta})^{\lambda_j}$ and retuen S_{ID_j} as the answer.

[Signcryption query on a plaintext m, a sender ID_i^*, and n arbitrary receivers $ID_{R_i} (i = 1, ..., n)$]: Note that $Q_{ID_i^*}$ has been set to $P^{\lambda_i^*} Q^{-1}$ for some $\lambda_i^* \in Z_p^*$. $\mathcal{B}$ first chooses $r', h_1, \lambda_R \in Z_p^*$ at random, computes $X = P^{r'} (P^{\beta})^{-h_1}$, $W = (P^{\lambda_i^*} Q^{-1})^{r'}$, $Q_R = P^{\lambda_R}$ and checks if L_1 already contains a tuple (m, X, Q_R, h_1') with $h_1' \neq h_1$. In this case, $\mathcal{B}$ repeats the process with another (r', h_1, λ_R) until finding a tuple (m, X, Q_R, h_1) whose first three elements do not figure in a tuple of L_1. Then B adds the admissible tuple into L_1, retrieves $(ID_{R_i}, \lambda_{R_i})$ from L_0 and computes $U_i = X^{(\lambda_R + \lambda_{R_i})}$ for $i = 1, ..., n$. $\mathcal{B}$ also computes $V = e(X, (P^{\beta})^{\lambda_R})$. Finally, $\mathcal{B}$ proceeds as in the normal signcryption process to produce the desired ciphertext σ.

[De-signcryption query]: when $\mathcal{A}$ submits a ciphertext $C = (X, Y, Z, Q_R, U_1, ..., U_n)$ and ID_i^* for some $i \in [1, n]$, $\mathcal{B}$ searches all combinations $(ID_{S,i}, m_i, X, W_i)$ such that $(m_i, X, Q_R, h_{1,i}) \in L_1$, $(V_i, h_{2,i}) \in L_2$, $(V_i, ID_{S,i}, h_{3,i}) \in L_3$, for some $h_{1,i}$, $h_{2,i}$, $h_{3,i}$, V_i under the constraints that $Y \cdot h_{3,i}^{-1} = W_i$, $h_{2,i} \oplus Z = ID_{S,i} || m_i$. If no such combination exists, the $\perp$ symbol is returned to signal that the ciphertext is invalid. Otherwise, all the tuples $(ID_{S,i}, m_i, X, W_i, h_{1,i})$ which satisfy the above constraints are kept for future examination. If one of them satisfies $e(P, W_i) = e\left(XP_{TA}^{h_{1,i}}, Q_{ID_{s,i}}\right)$, then $(ID_{S,i}, m_i)$ is returned.

Challenge: $\mathcal{A}$ outputs two messages m_0, m_1 together with an arbitrary sender's private key S_{ID_S} on which he wishes to be challenged. $\mathcal{B}$ searches the list L_0 to get λ_i^* that corresponds to ID_i^* for $i = 1, ..., n$. Then $\mathcal{B}$ responds with challenge ciphertext $c' = (X', Y', Z', U_1', ..., U_n', Q_R')$ built under the n target identities, where $X' = P^\alpha$, $Q_R' = Q$, $U_i' = X^{\lambda_i^*}$ for $i = 1, ..., n$, Y' and Z' are random strings of appropriate size.

Phase 2: $\mathcal{A}$ performs new queries as in Phase 1. However, it can not ask the de-signcryption query of the challenge c' with the private key of any target identity nor query the de-signcryption oracle on an identity and a ciphertext c which is only different from c' in the receiver information part.

At the end of the game, $\mathcal{A}$ returns its guess. $\mathcal{B}$ ignores the answer, randomly picks an entry (V, h_2) in L_2, and returns V as the solution to the BDH problem.

Analysis: For a signcryption query on a plaintext m and a sender ID_i^*, X is set to $P^{r'}(P^\beta)^{-h_1} = P^{r'-\beta h_1}$, so r is implicitly defined to $r' - \beta h_1$. Since $W = (P^{\lambda_i^*} Q^{-1})^{r'} = (P^{\lambda_i^*} Q^{-1})^{r'-\beta h_1} (P^{\lambda_i^*} Q^{-1})^{\beta h_1} = (Q_{ID_i^*})^{r'-\beta h_1} (Q_{ID_i^*})^{\beta h_1} = (Q_{ID_i^*})^r (S_{ID_i^*})^{h_1}$ and $U_i = X^{(\lambda_R + \lambda_{R_i})} = (P^r)^{(\lambda_R + \lambda_{R_i})} = (P^{\lambda_R} P^{\lambda_{R_i}})^r = (Q_R Q_{ID_{R_i}})^r$ for $i = 1, ..., n$, the distribution of the simulated target ciphertext is identical to that of the target ciphertext in the real attack.

For the challenge, we set $X' = P^\alpha$, $Q_R' = Q$ and $U_i' = X'^{\lambda_i^*}$ for $i = 1, ..., n$. Given $Q_{ID_i^*} = H_0(ID_i^*) = P^{\lambda_i^*} Q^{-1}$, we have $U_i' = X'^{\lambda_i^*} = P^{\alpha \lambda_i^*} = (P^{\lambda_i^*} Q^{-1} Q)^\alpha = (Q_{ID_i^*} Q_R')^\alpha$. Thus c' is identically distributed as that in the real attack. If $\mathcal{A}$ guesses correctly, it needs to query the random oracle H_2 with $V' = e(P_{TA}^r, Q_R') = e((P^\beta)^\alpha, Q) = e(P, Q)^{\alpha\beta}$. Then an entry (V', h_2) will be left in L_2, from which $\mathcal{B}$ can extract $e(P, Q)^{\alpha\beta}$.

Following the above discussion we know that, as long as the simulation of the attacker's environment is perfect, the probability that $\mathcal{A}$ asks the hash value of $e(P, Q)^{\alpha\beta}$ is the same as in a real attack. Now, we assess the probability that the simulation is not perfect. The only case where it can happen is when a valid ciphertext is rejected in a de-signcryption query. It is easy to see that for every $(V_i, h_{2,i})$ of L_2, there is exactly one pair $(h_{1,i}, h_{3,i})$ of elements in the range of oracles H_1 and H_3 providing a valid ciphertext. Thus the probability to reject a valid ciphertext is not greater than $q_{H_2}/2^{2k}$. Since $\mathcal{A}$ makes total q_{DSC} de-signcryption queries during the attack and $\mathcal{B}$ randomly chooses V from list L_2 as a solution of the BDH problem, we have $\epsilon' \geq \frac{1}{q_{H_2}} \left(\epsilon - q_{H_2} q_{DSC} / 2^{2k}\right)$. Moreover,

the bound on $\mathcal{B}$'s computation time derives from the fact that every signcryption query requires one pairing evaluation and every de-signcryption query requires two pairing evaluations.

Theorem 2. *In the random oracle, if a forger $\mathcal{F}$ has non-negligible advantage $\epsilon \geq 10n(q_{SC}+1)(q_{SC}+q_{H_1})/q + q_{H_2}q_{DSC}/2^{2k}$ against the SUF-sMIBSC-CMA security of our scheme when running in time t and performing q_{SC} signcryption queries, q_{DSC} de-signcryption queries and q_{H_i} queries to oracles H_i(for i=0,1,2,3), then there is an algorithm $\mathcal{B}$ that solves the CDH problem with probability $\epsilon_{\mathcal{B}} \geq 1/9$ and within running time $t_{\mathcal{B}} \leq 23nq_{H_1}(t+q_{SC}t_e)/(\epsilon - q_{H_2}q_{DSC}/2^{2k})$ where t_e denotes the time required for one pairing evaluation.*

We show how to build an algorithm $\mathcal{B}$ that solves the CDH problem. On input $(P, P^{\beta}, Q = P^{\gamma})$, $\mathcal{B}$'s goal is to compute $P^{\beta\gamma}$ or Q^{β}. Since it is difficult for $\mathcal{B}$ to directly make use of $\mathcal{F}$, we consider the following game, which is a variant of the game defined in Definition 5:

Considering that the most general known attack to signature schemes is chosen message attack, in this game the forger is not allowed to access de-signcryption oracle. However, he can still query all the other oracles: private key extraction oracle, signcryption oracle and hash oracles. In the beginning of the game, the forger is still assumed to output multiple target identities but he must commit to one of them. And in step "Forgery", the forger can not win the game if the outputted ciphertext is not a valid forged ciphertext for which the sender is the committed one. The first step of our proof is to reduce the problem to this new game.

Lemma 1. *If there is a forger $\mathcal{F}$ that has non-negligible advantage ϵ and running time t against the SUF-sMIBSC-CMA security of our scheme, then there exists a forger $\mathcal{F}'$ that wins the above game within running time $t' \leq t$ and with probability $\epsilon' \leq \frac{1}{n}(\epsilon - q_{H_2}q_{DSC}/2^{2k})$ where q_{DSC} and q_{H_2} are the numbers of de-signcryption queries and queries to H_2 issued by $\mathcal{F}$ respectively.*

Here we remark that the forger $\mathcal{F}'$ can be viewed as an adversary to the non-ID-based scheme obtained by fixing the identities of the sender and multiple receivers in our multi-receiver ID-based signcryption sheme. Assuming the existence of $\mathcal{F}'$, we can construct an algorithm which solves the CDH problem by rewinding $\mathcal{F}'$.

Lemma 2. *If there is a forger $\mathcal{F}'$ that wins the above variant game with advantage $\epsilon' \geq 10(q_{SC}+1)(q_{SC}+q_{H_1})/q$ when running in time t' and performing q_{SC} eigncryption queries, q_{H_i} queries to oracles H_i(for $i = 0, 1, 2, 3$), then there exists an algorithm $\mathcal{B}$ that solves the CDH problem with probability $\epsilon_{\mathcal{B}} \geq 1/9$ and within running time $t_{\mathcal{B}} \leq 23q_{H_1}(t'+q_{SC}t_e)/\epsilon'$ where t_e denotes the time required for one pairing evaluation.*

From the above two lemmas, it is easy to verify that Theorem 2 holds.

6 Comparison and Efficiency Discussion

First we present a possible multi-receiver ID based signcryption construction and discuss its security. Recall that in 2005, Baek et al. present an efficient multi-receiver identity based encryption scheme for which each encryption to n receivers also only requires one pairing computation. By adding sender authentication to Baek's scheme, we can construct a multi-receiver scheme which still maintains the efficiency feature. However, it may not be a secure one. For example, considering that Baek's scheme is a multi-receiver variant of IBE [14], we can use the similar method as Malone-Lee whose ID based signcryption scheme [10] is a result of a combination of IBE with a signature scheme to construct a MIBSC scheme. However, the combined scheme can not achieve the semantical security. As pointed put in [15], as soon as the signature on the plaintext is visible in the ciphertext, any attacker can simply verify the signature on plaintexts m_0 and m_1 produced during the game which is defined for the confidentiality and then find out which one matches to the challenge ciphertext. Similarly, other combination methods also suffer from certain security weakness.

Now we compare the efficiency of our method with that of several established ID based signcryption schemes. Since computation time and ciphertext size are two important factors affecting the efficiency, we present the comparison with respect to them. For comparison on computation time, we only consider the numbers of pairing computation as they are the most expensive. For comparison on ciphertext size, we use $||G_1||$, $||m||$, $||ID||$ to denote the size of an element in G_1, the length of message m and the length of identity ID respectively.

To make the comparison convincing, we consider two typical schemes. The first one is proposed by Boyen [8]. As the author pointed out, his scheme supports multi-recipient encryption with signature sharing for maximum scalability. This goal is achieved by carrying out the Sign operation of the scheme once, and then performing the Encrypt operation independently for each receiver, based on the output from Sign. However, as described in [8], each Encrypt operation requires one pairing computation, so n pairing computations are needed to send a message to n receivers. Besides, the ciphertext is in the form of $c = (\langle x_1, y_1 \rangle, ..., \langle x_n, y_n \rangle, z)$ where $\langle x_i, y_i \rangle \in G_1 \times G_1$ and the bit length of z is equal to the sum of $||ID||$ and $||m||$. So the ciphertext size is $2n||G_1|| + ||ID|| + ||m||$.

The second scheme was presented by Chen and Malone-Lee [12] and was believed to be the most efficient of its type proposed to date. Because this scheme is in the standard single-receiver setting, we apply the technique of randomness re-use to obtain the corresponding multi-receive scheme. And for the space limit, we do not provide the detail of the resulting scheme which will be available in the full version. But we mention that the total number of paring computations is equal to that of the receivers. In addition, the ciphertext is in the form of $(X, y_1, \ldots, y_n)$, where $X \in G_1$ and the bitlength of y_i is equal to the sum of $||G_1||$, $||ID||$ and $||m||$. So the ciphertext size is $||G_1|| + n(||G_1|| + ||ID|| + ||m||)$.

The comparisons are summarized in the following table.

Scheme	Ciphertext Size	Signcrypt Time
Our scheme	$(n+2)\|\|G_1\| + \|\|m\|\| + \|\|ID\|\|$	1
Boyen's scheme	$2n\|\|G_1\| + \|\|m\|\| + \|\|ID\|\|$	n
Chen's scheme	$(n+1)\|\|G_1\| + n\|\|ID\|\| + n\|\|m\|\|$	n

References

[1] M. Bellare, A. Boldyreva, S. Micali, Public-key encryption in a multi-user setting: Security proofs and improvements, in: B. Preneel (Ed.), Advances in Cryptology – EUROCRYPT ' 2000, Vol. 1807 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Germany, Brugge, Belgium, 2000, pp. 259–274.

[2] O. Baudron, D. Pointcheval, J. Stern, Extended notions of security for multicast public key cryptosystems, in: U. Montanari, J. D. P. Rolim, E. Welzl (Eds.), Proceedings of the 27th International Colloquium on Automata, Languages and Programming, ICALP'2000 (Geneva, Switzerland, July 9-15, 2000), Vol. 1853 of LNCS, Springer-Verlag, 2000, pp. 499–511.

[3] K. Kurosawa, Multi-recipient public-key encryption with shortened ciphertext, in: Public Key Cryptography, 2002, pp. 48–63.

[4] M. Bellare, A. Boldyreva, J. Staddon, Randomness re-use in multi-recipient encryption scheme, in: Public Key Cryptography, 2003, pp. 85–99.

[5] A. Shamir, Identity-based cryptosystem and signature scheme, in: G. R. Blakley, D. Chaum (Eds.), Advances in Cryptology – CRYPTO ' 84, Vol. 196 of Lecture Notes in Computer Science, International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1985, pp. 120–126.

[6] J. Baek, R. Safavi-Naini, W. Susilo, Efficient multi-receiver identity-based encryption and its application to broadcast encryption, in: Public Key Cryptography, 2005, pp. 380–397.

[7] Y. Zheng, Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption), in: CRYPTO, 1997, pp. 165–179.

[8] X. Boyen, Multipurpose identity-based signcryption (A swiss army knife for identity-based cryptography), in: CRYPTO, 2003, pp. 383–399.

[9] M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in: ACM Conference on Computer and Communications Security, 1993, pp. 62–73.

[10] J. Malone-lee, Identity-based signcryption (Jul. 19 2002).
URL `http://eprint.iacr.org/2002/098.ps.gz`

[11] J. jacques Quisquater, New identity based signcryption schemes from pairings (Feb. 24 2003).
URL `http://eprint.iacr.org/2003/023.ps.gz`

[12] L. Chen, J. Malone-Lee, Improved identity-based signcryption, in: Public Key Cryptography, 2005, pp. 362–379.

[13] R. Canetti, S. Halevi, J. Katz, A forward-secure public-key encryption scheme, in: EUROCRYPT, 2003, pp. 255–271.

[14] D. Boneh, M. K. Franklin, Identity-based encryption from the weil pairing, SIAM J. Comput 32 (3) (2003) 586–615.

[15] J.-B. Shin, K. Lee, K. Shim, New DSA-verifiable signcryption schemes, in: ICISC, 2002, pp. 35–47.

Efficient Identity-Based Signatures Secure in the Standard Model*

Kenneth G. Paterson and Jacob C.N. Schuldt

Information Security Group,
Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, UK
{kenny.paterson, jacob.schuldt}@rhul.ac.uk

Abstract. The only known construction of identity-based signatures that can be proven secure in the standard model is based on the approach of attaching certificates to non-identity-based signatures. This folklore construction method leads to schemes that are somewhat inefficient and leaves open the problem of finding more efficient direct constructions. We present the first such construction. Our scheme is obtained from a modification of Waters' recently proposed identity-based encryption scheme. It is computationally efficient and the signatures are short. The scheme's security is proven in the standard model and rests on the hardness of the computational Diffie-Hellman problem in groups equipped with a pairing.

1 Introduction

Identity based encryption (IBE), introduced by Shamir [Sha84], enables the computation of a public key for an entity, given only some general scheme parameters and a string identifying the entity (e.g. an e-mail address, a telephone number, etc.). A private key generator (PKG) computes private keys from a master secret and distributes these to the entities participating in the scheme. This eliminates the need for certificates as used in a traditional public key infrastructure. Although Shamir proposed the idea of an IBE scheme in 1984, no construction that was both efficient and secure was found until recently, when the work of Boneh and Franklin [BF01] and Cocks [Coc01] was published. Since then, a large number of papers have been published in this area (see [Bar] for a list of some of these), including several containing direct constructions of identity-based signature (IBS) schemes [Pat02, Hes02, CC03, Yi03, BLMQ05].

Most of these IBS schemes are provably secure in the random oracle model [BR93]. However, it has been shown that when random oracles are instantiated with concrete hash functions, the resulting scheme may not be secure [CGH98, BBP04]. Recently, efforts have been made to construct IBE schemes that are provably secure in the standard model to overcome this problem. Boneh

* An extended version of this paper is available at http://eprint.iacr.org/2006/080.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 207–222, 2006.
© Springer-Verlag Berlin Heidelberg 2006

and Boyen initially proposed an IBE scheme [BB04a] which could be proven secure using a "Selective ID" security model that is slightly weaker than the model original proposed by Boneh and Franklin. The same authors later proposed an IBE scheme [BB04b] that is secure in the full Boneh-Franklin security model and in the standard model, but which is inefficient. Finally, Waters [Wat05] succeeded in constructing a fairly efficient IBE scheme which meets both requirements. Naccache [Nac05] and Chatterjee-Sarkar [SC05] independently proposed a technique for reducing the size requirements of Waters' scheme, making it more suitable for practical use.

A generic approach to the construction of IBS schemes is to use an ordinary (i.e. non-identity-based) signature scheme and simply attach a certificate containing the public key of the signer to the signature. This certification-based approach is apparently folklore, and variants of it are mentioned in passing in several recent papers [GS02, DKXY03, BNN04, KMPR05]. The simplicity of this technique shows that, in some sense, IBS schemes are much easier to obtain than IBE schemes. Moreover, using an ordinary signature scheme that is secure in the standard model as a component in the construction, one can obtain an IBS scheme that is also secure in the standard model. However, this construction does have its disadvantages. One is that the resulting identity-based signature will need to include both a public key (that of the signer) and two ordinary signatures (one from the signer and one from the certifier). This impacts on the length of the signatures. Another is that each signature verification in the identity-based scheme will involve verification of two ordinary signatures. These drawbacks lead to somewhat inefficient schemes and raise the question of whether it is possible to do better in the standard model with a direct construction.

An interesting observation, attributed to Naor by Boneh and Franklin, is that any IBE scheme can be used to construct an ordinary signature scheme. This is done by keeping the master secret of the IBE scheme as the private key and publishing the scheme parameters of the IBE scheme as the public key. A signature on a message $\mathfrak{m}$ is then the private key of the identity $\mathfrak{u}_m = \mathfrak{m}$ and verification can be performed by selecting a random message $\mathfrak{m}_r$, encrypting $\mathfrak{m}_r$ with the public key of $\mathfrak{u}_m$, and verifying that decryption is possible using the given signature as a decryption key. If the used IBE scheme is IND-ID-CPA secure, the resulting signature scheme is existentially unforgeable under an adaptive chosen message attack. This technique was used by Boneh, Lynn and Shacham [BLS04] to construct short signatures from the IBE scheme of Boneh and Franklin [BF01], and likewise by Boneh and Boyen to obtain another short signature scheme [BB04c] from an IBE scheme due to the same authors [BB04a]. As noted by Gentry and Silverberg [GS02], IBS schemes can be constructed in a very similar way if a hierarchical IBE (HIBE) scheme is used in place of an IBE scheme. This will, in fact, lead to a hierarchical IBS scheme where signing identities are part of a hierarchy having one level less than the used HIBE scheme. When an identity $(\mathfrak{u}_1, \ldots, \mathfrak{u}_t)$ signs a message $\mathfrak{m}$, the identity $\mathfrak{u}_m = \mathfrak{m}$ is inserted as a child of $(\mathfrak{u}_1, \ldots, \mathfrak{u}_t)$ in the hierarchy. As above, a signature is the private key of $\mathfrak{u}_m$ and a verifier checks that decryption of a random message, encrypted

with the public key for identity $(\mathtt{u}_1, \ldots, \mathtt{u}_t, \mathtt{u}_m)$, is possible. Limiting the used HIBE to a 2-level scheme leads to an ordinary IBS.

Our contribution. As a natural extension of the efforts to provide secure schemes without the use of random oracles, we give the first direct construction for an IBS scheme that is provably secure in the standard model. Our scheme is based on a hierarchical extension of Waters' scheme, and we use the above-described technique of converting a 2-level HIBE scheme into a IBS scheme for our construction. We prove our signature scheme to be secure under the computational Diffie-Hellman assumption. This assumption seems more natural than many of the hardness assumptions recently introduced to pairing based cryptography.

In terms of signature size and computational cost, our new scheme is competitive with existing identity-based signature schemes (that are provably secure only in the random oracle model). Our signatures consist of 3 group elements, while signing is pairing-free and verification needs 3 pairing computations. Our scheme is more efficient in terms of computation and signature size than the IBS scheme that results from applying the certification-based construction to Waters' signature scheme. In comparison to the IBS scheme that can be obtained from the signature scheme of Boneh and Boyen [BB04c], our signatures are roughly half the size, but verification takes one more pairing computation. However, security for the Boneh-Boyen-derived scheme depends on the q-SDH assumption, whereas the security of our scheme rests on the arguably more appealing computational Diffie-Hellman assumption.

The only drawback of our scheme is the relatively large size of its public parameters. However, we show how the technique of Naccache and Chatterjee-Sarkar can be applied to our scheme to reduce the size of the public parameters, at the cost of a looser security reduction.

2 Identity-Based Signatures

An identity-based signatures scheme can be described as a collection of the following four algorithms:

Setup. This algorithm is run by the master entity on input a security parameter, and generates the public parameters `params` of the scheme and a master secret. The master entity publishes `params` and keeps the master secret to itself.

Extract. Given an identity $\mathtt{u}$, the master secret and `params`, this algorithm generates the private key $d_{\mathtt{u}}$ of $\mathtt{u}$. The master entity will use this algorithm to generate private keys for all entities participating in the scheme and distribute the private keys to their respective owners through a secure channel.

Sign. Given a message $\mathtt{m}$, an identity $\mathtt{u}$, a private key $d_{\mathtt{u}}$ and `params`, this algorithm generates the signature σ of $\mathtt{u}$ on $\mathtt{m}$. The entity with identity $\mathtt{u}$ will use this algorithm for signing.

Verify. Given a signature σ, a message $\mathtt{m}$, an identity $\mathtt{u}$ and `params`, this algorithm outputs `accept` if σ is a valid signature on $\mathtt{m}$ for identity $\mathtt{u}$, and outputs `reject` otherwise.

2.1 Existential Unforgeability

The security model of existential unforgeability under an adaptive chosen message attack, defined by Goldwasser, Micali and Rivest [GMR88], can be extended to the identity-based scenario in a natural way. We will define security for identity-based signature schemes by the following game between a challenger and an adversary:

Setup. The challenger runs the algorithm **Setup** of the signature scheme and obtains both the public parameters $\mathtt{params}$ and the master secret. The adversary is given $\mathtt{params}$ but the master secret is kept by the challenger.

Queries. The adversary adaptively makes a number of different queries to the challenger. Each query can be one of the following.

- Extract query. The adversary can ask for the private key of any identity $\mathtt{u}$. The challenger responds by running **Extract**$(\mathtt{params}, \mathtt{u})$ and forwards the private key $d_{\mathtt{u}}$ to the adversary.
- Sign query. The adversary can ask for the signature of any identity $\mathtt{u}$ on any message $\mathtt{m}$. The challenger responds by first running **Extract** $(\mathtt{params}, \mathtt{u})$ to obtain the private key $d_{\mathtt{u}}$ of $\mathtt{u}$, and then running **Sign** $(\mathtt{params}, d_{\mathtt{u}}, \mathtt{u}, \mathtt{m})$ to obtain a signature, which is forwarded to the adversary.

Forgery. The adversary outputs a message $\mathtt{m}^*$, an identity $\mathtt{u}^*$ and a string σ^*. The adversary *succeeds* if the following hold true:

1. **Verify**$(\mathtt{params}, \mathtt{u}^*, \mathtt{m}^*, \sigma^*) = \mathtt{accept}$
2. The adversary has not made an extract query on $\mathtt{u}^*$.
3. The adversary has not made a sign query on $(\mathtt{u}^*, \mathtt{m}^*)$.

The advantage of an adversary $\mathcal{A}$ in the above game is defined to be

$$\mathrm{Adv}_{\mathcal{A}} = Pr[\mathcal{A} \textit{ succeeds}]$$

where the probability is taken over all coin tosses made by the challenger and the adversary.

Definition 1. *An adversary $\mathcal{A}$ is said to be an (ϵ, t, q_e, q_s)-forger of an identity-based signature scheme if $\mathcal{A}$ has advantage at least ϵ in the above game, runs in time at most t, and makes at most q_e and q_s extract and sign queries, respectively. A scheme is said to be (ϵ, t, q_e, q_s)-secure if no (ϵ, t, q_e, q_s)-forger exists.*

The above game can easily be extended to cover strong unforgeability [ADR02] by changing the third requirement in the forgery stage to "σ^* was not output as a response to a sign query". However, our concrete scheme does not enjoy security in this stronger sense, as an adversary can easily modify an existing signature on a message into a new signature on the same message.

3 Complexity Assumptions

The security of our signature scheme will be reduced to the hardness of the computational Diffie-Hellman (CDH) problem in the group in which the signature is constructed. We briefly review the definition of the CDH problem:

Definition 2. *Given a group $\mathbb{G}$ of prime order p with generator g and elements $g^a, g^b \in \mathbb{G}$ where a, b are selected uniformly at random from $\mathbb{Z}_p^*$, the CDH problem in $\mathbb{G}$ is to compute g^{ab}.*

Definition 3. *We say that the (ϵ, t)-CDH assumption holds in a group $\mathbb{G}$ if no algorithm running in time at most t can solve the CDH problem in $\mathbb{G}$ with probability at least ϵ.*

4 Construction

Our new identity-based signature scheme is based on an hierarchical extension of the identity-based encryption scheme presented by Waters [Wat05]. However, as shown in Section 5, the security of our scheme can be reduced to the hardness of the CDH problem, whereas Waters' scheme relies on the stronger Bilinear Diffie-Hellman assumption. Our construction is based on bilinear maps and we now briefly review some of the basic properties of such maps.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p and let g be a generator of $\mathbb{G}$. The map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is said to be an admissible map if the following three conditions hold true:

- e is bilinear, i.e. $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
- e is non-degenerate, i.e. $e(g, g) \neq 1$.
- e is efficiently computable.

See [Gal05] for more details on the construction of such maps. In Section 7, we will sketch the modifications necessary to allow our scheme to operate in the more general setting where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with $\mathbb{G}_1 \neq \mathbb{G}_2$.

In the following all identities and messages will be assumed to be bit strings of length n_u and n_m, respectively. To construct a more flexible scheme which allows identities and messages of arbitrary lengths, collision-resistant hash functions, $H_u : \{0,1\}^* \to \{0,1\}^{n_u}$ and $H_m : \{0,1\}^* \to \{0,1\}^{n_m}$, can be defined and used to create identities and messages of the desired length. We will use the notation $v \leftarrow_R S$ as a short hand for choosing a value v uniformly at random from the set S. Our new signature scheme is defined by the following algorithms:

Setup. Choose groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p such that an admissible pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ can be constructed and pick a generator g of $\mathbb{G}$.
Now, pick a secret $\alpha \leftarrow_R \mathbb{Z}_p$, compute $g_1 = g^\alpha$ and pick $g_2 \leftarrow_R \mathbb{G}$. Furthermore, pick elements $u', m' \leftarrow_R \mathbb{G}$ and vectors $\boldsymbol{U} = (u_i)$, $\boldsymbol{M} = (m_i)$ of length n_u and n_m, respectively, whose entries are random elements from $\mathbb{G}$. The public parameters are $\texttt{params} = (\mathbb{G}, \mathbb{G}_T, e, g, g_1, g_2, u', \boldsymbol{U}, m', \boldsymbol{M})$ and the master secret is g_2^α.

Extract. Let $\mathfrak{u}$ be a bit string of length n_u representing an identity and let $\mathfrak{u}[i]$ be the ith bit of $\mathfrak{u}$. Define $\mathcal{U} \subset \{1, \ldots, n_u\}$ to be the set of indicies i such that $\mathfrak{u}[i] = 1$.

To construct the private key, $d_{\mathfrak{u}}$, of the identity $\mathfrak{u}$, pick $r_u \leftarrow_R \mathbb{Z}_p$ and compute:

$$d_{\mathfrak{u}} = \left(g_2^{\alpha} \left(u' \prod_{i \in \mathcal{U}} u_i \right)^{r_u}, g^{r_u} \right).$$

Note that a user can easily re-randomize his private key after he has received it from the master entity.

Sign. Let $\mathfrak{u}$ be the bit string of length n_u representing a signing identity and let $\mathfrak{m}$ be a bit string representing a message. As in the **Extract** algorithm, let $\mathcal{U}$ be the set of indicies i such that $\mathfrak{u}[i] = 1$, and likewise, let $\mathcal{M} \subset \{1, \ldots, n_m\}$ be the set of indicies j such that $\mathfrak{m}[j] = 1$, where $\mathfrak{m}[j]$ is the jth bit of $\mathfrak{m}$.

A signature of $\mathfrak{u}$ on $\mathfrak{m}$ is constructed by picking $r_m \leftarrow_R \mathbb{Z}_p$ and computing

$$\sigma = \left(g_2^{\alpha} \left(u' \prod_{i \in \mathcal{U}} u_i \right)^{r_u} \left(m' \prod_{j \in \mathcal{M}} m_j \right)^{r_m}, g^{r_u}, g^{r_m} \right) \in \mathbb{G}^3.$$

Verify. Given a purported signature $\sigma = (V, R_u, R_m) \in \mathbb{G}^3$ of an identity $\mathfrak{u}$ on a message $\mathfrak{m}$, a verifier accepts σ if the following equality holds:

$$e(V, g) = e(g_2, g_1) e\left(u' \prod_{i \in \mathcal{U}} u_i, R_u \right) e\left(m' \prod_{j \in \mathcal{M}} m_j, R_m \right).$$

It is easy to see that a signature constructed with the **Sign** algorithm will be accepted by a verifier. Thus the scheme is correct.

4.1 Efficiency

Our scheme has a private key size and a signature size of two and three group elements, respectively. Note, however, that the second value of a signature tuple, g^{r_u}, will remain the same for all signatures made by a given user. Hence, if many messages are signed by a single user and verified by a single verifier, the value g^{r_u} will only need to be included in one of the signatures. The public parameters of our scheme will consist of a description of the groups $\mathbb{G}, \mathbb{G}_T$ and the pairing e, and $n_u + n_m + 5$ group elements of $\mathbb{G}$. In a practical scheme, the size of the public parameters will be a performance concern and in Section 6 we will discuss how the number of group elements needed in `params` can be reduced.

To construct a signature, a signer will need to compute at most $n_m + 1$ multiplications in $\mathbb{G}$ ($n_m/2 + 1$ on average) and perform two exponentiations in $\mathbb{G}$. Verification requires at most $n_u + n_m$ multiplications in $\mathbb{G}$ ($(n_u + n_m)/2$ on average) and four pairing computations. However, the value $e(g_1, g_2)$ can be precomputed and cached, reducing the verification cost by one pairing. A further pairing can be eliminated if a verifier checks multiple signatures from a single signer.

Thus, our scheme is only slightly more expensive than existing IBS schemes (see for example the table in [Hes02]). However, these schemes are only proven secure in the random oracle model while our scheme, as the next section will show, can be proven secure in the standard model. We have already compared our scheme to IBS schemes that result from the certification-based construction applied to standard-model-secure schemes in the introduction.

5 Proof of Security

We will prove that our identity-based signature scheme is existentially unforgeable under a chosen message attack, in the standard model, given that the computational Diffie-Hellman problem is hard.

Theorem 1. *The identity-based signature scheme of Section 4 is* (ϵ, t, q_e, q_s)*-secure, assuming that the* (ϵ', t')*-CDH assumption holds in* $\mathbb{G}$*, where:*

$$\epsilon' = \frac{\epsilon}{16(q_e + q_s)q_s(n_u + 1)(n_m + 1)},$$
$$t' = t + O\big((q_e n_u + q_s(n_u + n_m))\rho + (q_e + q_s)\tau\big),$$

and ρ *and* τ *are the time for a multiplication and an exponentiation in* $\mathbb{G}$*, respectively.*

Proof. We will assume that an (ϵ, t, q_e, q_s)-forger $\mathcal{A}$ for our scheme exists. From this forger, we will construct an algorithm $\mathcal{B}$ that solves CDH with probability at least ϵ' and in time at most t', contradicting the (ϵ', t')-CDH assumption. Our approach is based on that of [Wat05].

The algorithm $\mathcal{B}$ will be given a group $\mathbb{G}$, a generator g and the elements g^a and g^b. To be able to use $\mathcal{A}$ to compute g^{ab}, $\mathcal{B}$ must be able to simulate a challenger for $\mathcal{A}$. Such a simulation can be created in the following way:

Setup. $\mathcal{B}$ sets $l_u = 2(q_e + q_s)$ and $l_m = 2q_s$, and randomly chooses two integers k_u and k_m, with $0 \le k_u \le n_u$ and $0 \le k_m \le n_m$. We will assume that $l_u(n_u + 1) < p$ and $l_m(n_m + 1) < p$ for the given values of q_e, q_s, n_u and n_m. The simulator then chooses an integer $x' \leftarrow_R \mathbb{Z}_{l_u}$ and a vector $\boldsymbol{X} = (x_i)$ of length n_u, with $x_i \leftarrow_R \mathbb{Z}_{l_u}$ for all i. Likewise, it randomly chooses another integer $z' \leftarrow_R \mathbb{Z}_{l_m}$ and a vector $\boldsymbol{Z} = (z_j)$ of length n_m, with $z_j \leftarrow_R \mathbb{Z}_{l_m}$ for all j. Lastly, $\mathcal{B}$ chooses two integers $y', w' \leftarrow_R \mathbb{Z}_p$ and two vectors, $\boldsymbol{Y} = (y_i)$ and $\boldsymbol{W} = (w_j)$, of length n_u and n_m, respectively, with $y_i, w_j \leftarrow_R \mathbb{Z}_p$ for all i and j.

To make the notation easier to follow, the following two pairs of functions are defined for an identity $\mathfrak{u}$ and a message $\mathfrak{m}$ respectively:

$$F(\mathfrak{u}) = x' + \sum_{i \in \mathcal{U}} x_i - l_u k_u \quad \text{and} \quad J(\mathfrak{u}) = y' + \sum_{i \in \mathcal{U}} y_i,$$
$$K(\mathfrak{m}) = z' + \sum_{j \in \mathcal{M}} z_j - l_m k_m \quad \text{and} \quad L(\mathfrak{m}) = w' + \sum_{j \in \mathcal{M}} w_j$$

Now, $\mathcal{B}$ constructs a set of public parameters for the IBE scheme by making the following assignments:

$$\begin{array}{ll} g_1 = g^a, & g_2 = g^b \\ u' = g_2^{-l_u k_u + x'} g^{y'}, & u_i = g_2^{x_i} g^{y_i} \;\; 1 \le i \le n_u \\ m' = g_2^{-l_m k_m + z'} g^{w'}, & m_j = g_2^{z_j} g^{w_j} \;\; 1 \le j \le n_m \end{array}$$

Note that these public parameters will have the same distribution as in the game between the challenger and $\mathcal{A}$. Furthermore, this assignment means that the master secret will be $g_2^\alpha = g_2^a = g^{ab}$ and that for any identity $\mathfrak{u}$ and message $\mathfrak{m}$, the equations

$$u' \prod_{i \in \mathcal{U}} u_i = g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})} \quad \text{and} \quad m' \prod_{j \in \mathcal{M}} m_j = g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})}$$

hold. All public parameters are passed to $\mathcal{A}$.

Queries. When running the adversary, both extract and sign queries can occur. $\mathcal{B}$ answers these in the following way:

– Extract queries. Consider a query for the private key of an identity $\mathfrak{u}$. $\mathcal{B}$ does not know the master secret, but assuming $F(\mathfrak{u}) \neq 0 \mod p$, it can construct a private key by choosing $r_u \leftarrow_R \mathbb{Z}_p$ and computing:

$$d_{\mathfrak{u}} = (d_0, d_1) = \left(g_1^{-J(\mathfrak{u})/F(\mathfrak{u})} \left(u' \prod_{i \in \mathcal{U}} u_i \right)^{r_u}, \; g_1^{-1/F(\mathfrak{u})} g^{r_u} \right)$$

Writing $\tilde{r}_u = r_u - a/F(v)$, it can be verified that defining $d_{\mathfrak{u}}$ in this manner yields a valid private key of $\mathfrak{u}$, since:

$$\begin{aligned} d_0 &= g_1^{-J(\mathfrak{u})/F(\mathfrak{u})} \left(u' \prod_{i \in \mathcal{U}} u_i \right)^{r_u} \\ &= g_2^a (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{-a/F(\mathfrak{u})} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} \\ &= g_2^a (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u - a/F(\mathfrak{u})} \\ &= g_2^a \left(u' \prod_{i \in \mathcal{U}} u_i \right)^{\tilde{r}_u} \end{aligned}$$

and

$$d_1 = g_1^{-1/F(\mathfrak{u})} g^{r_u} = g^{r_u - a/F(\mathfrak{u})} = g^{\tilde{r}_u}.$$

Hence, to the adversary, all private keys computed by $\mathcal{B}$ will be indistinguishable from the keys generated by a true challenger.

If, on the other hand, $F(\mathfrak{u}) = 0 \mod p$, the above computation cannot be performed and the simulator will abort. To make the analysis of the simulation easier, we will force the simulator to abort whenever $F(\mathfrak{u}) = 0 \mod l_u$. Given the assumption $l_u(n_u + 1) < p$ which implies

$0 \leq l_u k_u < p$ and $0 \leq x' + \sum_{i \in \mathcal{U}} x_i < p$, it is easy to see that $F(\mathfrak{u}) = 0 \mod p$ implies that $F(\mathfrak{u}) = 0 \mod l_u$. Hence, $F(\mathfrak{u}) \neq 0 \mod l_u$ implies $F(\mathfrak{u}) \neq 0 \mod p$, so the former condition will be a sufficient requirement to ensure that a private key for $\mathfrak{u}$ can be constructed.

– Sign queries. Consider a query for a signature of $\mathfrak{u}$ on $\mathfrak{m}$ (it can be assumed, without loss of generality, that $\mathcal{A}$ has not made an extraction query on $\mathfrak{u}$). If $F(\mathfrak{u}) \neq 0 \mod l_u$, $\mathcal{B}$ can just construct a private key for $\mathfrak{u}$ as in an extract query, and then use the **Sign** algorithm to create a signature on $\mathfrak{m}$.

If $F(\mathfrak{u}) = 0 \mod l_u$, $\mathcal{B}$ will try to construct a signature in a similar way to the construction of a private key in an extract query. Assume $K(\mathfrak{m}) \neq 0 \mod l_m$. Arguing as above, this implies $K(\mathfrak{m}) \neq 0 \mod p$, given the assumption $l_m(n_m + 1) < p$. The signature of $\mathfrak{u}$ on $\mathfrak{m}$ can now be constructed by picking $r_u, r_m \leftarrow_R \mathbb{Z}_p$ and computing

$$\sigma = \left(\left(u' \prod_{i \in \mathcal{U}} u_i \right)^{r_u} g_1^{-L(\mathfrak{m})/K(\mathfrak{m})} \left(m' \prod_{j \in \mathcal{M}} m_i \right)^{r_m}, g^{r_u}, g_1^{-1/K(\mathfrak{m})} g^{r_m} \right)$$
$$= \left(g_2^a \left(u' \prod_{i \in \mathcal{U}} u_i \right)^{r_u} \left(m' \prod_{j \in \mathcal{M}} m_j \right)^{\tilde{r}_m}, g^{r_u}, g^{\tilde{r}_m} \right),$$

where $\tilde{r}_m = r_m - a/K(\mathfrak{m})$. This last equation shows that $\mathcal{B}$'s replies to $\mathcal{A}$'s sign queries are distributed as they would be in an interaction with a real challenger.

If $K(\mathfrak{m}) = 0 \mod l_m$, the simulator will simply abort.

Forgery. If $\mathcal{B}$ does not abort as a consequence of one of the queries above, $\mathcal{A}$ will, with probability at least ϵ, return an identity $\mathfrak{u}^*$, a message $\mathfrak{m}^*$, and a valid forgery $\sigma^* = (V, R_u, R_m)$ of a signature of $\mathfrak{u}^*$ on $\mathfrak{m}^*$. If $F(\mathfrak{u}^*) \neq 0 \mod p$ or $K(\mathfrak{m}^*) \neq 0 \mod p$ then $\mathcal{B}$ will abort. If, on the other hand, $F(\mathfrak{u}^*) = 0 \mod p$ and $K(\mathfrak{m}^*) = 0 \mod p$, $\mathcal{B}$ computes and outputs

$$\frac{V}{R_u^{J(\mathfrak{u}^*)} R_m^{L(\mathfrak{m}^*)}} = \frac{g_2^a \left(u' \prod_{i \in \mathcal{U}} u_i \right)^{r_u} \left(m' \prod_{j \in \mathcal{M}} m_j \right)^{r_m}}{g^{J(\mathfrak{u}^*) r_u} g^{L(\mathfrak{m}^*) r_m}} = g^{ab}$$

which is the solution to the given CDH problem.

This completes the description of the simulation. It remains to analyse the probability of $\mathcal{B}$ not aborting. For the simulation to complete without aborting, we require that all extraction queries on an identity $\mathfrak{u}$ have $F(\mathfrak{u}) \neq 0 \mod l_u$, that all sign queries $(\mathfrak{u}, \mathfrak{m})$ will either have $F(\mathfrak{u}) \neq 0 \mod l_u$ or $K(\mathfrak{m}) \neq 0 \mod l_m$, and that $F(\mathfrak{u}^*) = 0 \mod l_u$ and $K(\mathfrak{m}^*) = 0 \mod l_m$. However, to make the analysis simpler, we will bound the probability of a subcase of this event. More specifically, we will divide the sign queries into two groups – queries involving $\mathfrak{u}^*$ and queries involving identities $\mathfrak{u} \neq \mathfrak{u}^*$ – and then consider the event that all identities $\mathfrak{u}$ have $F(\mathfrak{u}) \neq 0 \mod l_u$, ignoring that sign queries $(\mathfrak{u}, \mathfrak{m})$ can

be answered if $F(\mathfrak{u}) = 0 \mod l_u$ and $K(\mathfrak{m}) \neq 0 \mod l_m$. Thus we will provide a lower bound on the probability that $\mathcal{B}$ aborts.

Let $\mathfrak{u}_1, \ldots, \mathfrak{u}_{q_I}$ be the identities appearing in either extract queries or in sign queries not involving the challenge identity and let $\mathfrak{m}_1, \ldots, \mathfrak{m}_{q_M}$ be the messages in the sign queries involving the challenge identity $\mathfrak{u}^*$. Clearly, we will have $q_I \leq q_e + q_s$ and $q_M \leq q_s$. Define the events A_i, A^*, B_j and B^* as

$$\begin{aligned} A_i &: F(\mathfrak{u}_i) \neq 0 \mod l_u \\ A^* &: F(\mathfrak{u}^*) = 0 \mod p \\ B_j &: K(\mathfrak{m}_j) \neq 0 \mod l_m \\ B^* &: K(\mathfrak{m}^*) = 0 \mod p \end{aligned}$$

From the analysis above, the probability of $\mathcal{B}$ not aborting is

$$\Pr[\neg\mathtt{abort}] \geq \Pr[\bigwedge_{i=1}^{q_I} A_i \wedge A^* \wedge \bigwedge_{j=1}^{q_M} B_j \wedge B^*].$$

It is easy to see that the events $(\bigwedge_{i=1}^{q_I} A_i \wedge A^*)$ and $(\bigwedge_{j=1}^{q_M} B_j \wedge B^*)$ are independent. Essentially, this is because the functions F and K which define these events are selected independently and are hidden from the adversary's view of the simulation.

As seen above, the assumption $l_u(n_u+1) < p$ leads to the implication $F(\mathfrak{u}) = 0 \mod p \Rightarrow F(\mathfrak{u}) = 0 \mod l_u$. Furthermore, this assumption gives that if $F(\mathfrak{u}) = 0 \mod l_u$, there will be an unique choice of k_u with $0 \leq k_u \leq n_u$ such that $F(\mathfrak{u}) = 0 \mod p$. Since k_u and $x', \boldsymbol{X}$ are randomly chosen, we have

$$\begin{aligned} \Pr[A^*] &= \Pr[F(\mathfrak{u}^*) = 0 \mod p \wedge F(\mathfrak{u}^*) = 0 \mod l_u] \\ &= \Pr[F(\mathfrak{u}^*) = 0 \mod l_u] \Pr[F(\mathfrak{u}^*) = 0 \mod p | F(\mathfrak{u}^*) = 0 \mod l_u] \\ &= \frac{1}{l_u} \frac{1}{n_u + 1} \end{aligned}$$

We also have that

$$\begin{aligned} \Pr[\bigwedge_{i=1}^{q_I} A_i | A^*] &= 1 - \Pr[\bigvee_{i=1}^{q_I} \neg A_i | A^*] \\ &\geq 1 - \sum_{i=1}^{q_I} \Pr[\neg A_i | A^*] \end{aligned}$$

If F is evaluated on two different identities, $\mathfrak{u}_1$ and $\mathfrak{u}_2$, then the sums appearing in $F(\mathfrak{u}_1)$ and $F(\mathfrak{u}_2)$ will differ in at least one randomly chosen value, and the events $F(\mathfrak{u}_1) = 0 \mod l_u$ and $F(\mathfrak{u}_1) = 0 \mod l_u$ will be independent. As a special case, the events A_i and A^* are independent for any i, and $\Pr[\neg A_i | A^*] = 1/l_u$. Hence, we have

$$\begin{aligned} \Pr[\bigwedge_{i=1}^{q_I} A_i \wedge A^*] &= \Pr[A^*] \Pr[\bigwedge_{i=1}^{q_I} A_i | A^*] \\ &\geq \frac{1}{l_u(n_u+1)} \left(1 - \frac{q_e + q_s}{l_u}\right) \end{aligned}$$

and setting $l_u = 2(q_e + q_s)$ as in the simulation gives

$$\Pr[\bigwedge_{i=1}^{q_I} A_i \wedge A^*] \geq \frac{1}{4(q_e + q_s)(n_u + 1)}.$$

A similar analysis for the sign queries gives the result

$$\Pr[\bigwedge_{j=1}^{q_M} B_j \wedge B^*] \geq \frac{1}{4q_s(n_m + 1)}$$

and we get that

$$\begin{aligned}\Pr[\neg\mathtt{abort}] &\geq \Pr[\bigwedge_{i=1}^{q_I} A_i \wedge A^*] \Pr[\bigwedge_{j=1}^{q_M} B_j \wedge B^*] \\ &\geq \frac{1}{16(q_e + q_s)q_s(n_u + 1)(n_m + 1)}\end{aligned}$$

If the simulation does not abort, $\mathcal{A}$ will create a valid forgery with probability at least ϵ. The algorithm $\mathcal{B}$ can then compute g^{ab} from the forgery as shown above.

The time complexity of the algorithm $\mathcal{B}$ is dominated by the exponentiations and, for larger values of n_u and n_m, multiplications performed in the extract and sign queries. Since there are $O(n_u)$ and $O(n_u + n_m)$ multiplications and $O(1)$ and $O(1)$ exponentiations in the extract and sign stage respectively, the time complexity of $\mathcal{B}$ is $t + O\big((q_e n_u + q_s(n_u + n_m))\rho + (q_e + q_s)\tau\big)$. Thus, the theorem follows. □

6 Trading Security for Efficiency

In a practical scheme, the n_u-bit identities and the n_m-bit messages will most likely be outputs from collision resistant hash functions. This suggests that n_u and n_m should be at least 160, which means that the public parameters of our scheme will contain at least 325 group elements of $\mathbb{G}$. When $\mathbb{G}$ is chosen such that the CDH problem in $\mathbb{G}$ is considered to be hard, the public parameters will grow to a size which is not suitable for environments with limited storage capacity. However, Naccache [Nac05] and Chatterjee-Sarkar [SC05] independently suggested a modification to Waters' scheme to reduce the size of the public parameters. This modification is also applicable to our signature scheme.

6.1 The Technique of Naccache and Chatterjee-Sarkar

In our signature scheme, when an entity signs a message $\mathfrak{m} \subset \{0,1\}^{n_m}$, he computes the product

$$m' \prod_{j \in \mathcal{M}} m_j$$

where $\mathcal{M} \subset \{1, \ldots, n_m\}$ is the set of indicies j such that the jth bit of $\mathfrak{m}$ is 1. The idea is to consider the message as a set of concatenated t-bit integers instead

of a set of concatenated bits, i.e. $\mathfrak{m} = \mathfrak{m}[1]||\cdots||\mathfrak{m}[n'_m]$ where $n'_m = n_m/t$ and $\mathfrak{m}[j] \in \mathbb{Z}_{2^t}$, and then replace the above product with

$$m' \prod_{j=1}^{n'_m} m_j^{\mathfrak{m}[j]}.$$

This will reduce the size of $\boldsymbol{M}$ by a factor of $n_m/n'_m = t$ and the number of group elements included in the public parameters `params` will be reduced to $n_u + n_m/t + 5$.

Likewise, an identity $\mathfrak{u} \subset \{0,1\}^{n_u}$ can be considered as a concatenation of n'_u s-bit integers and by replacing the product of elements from $\boldsymbol{U}$ in a similar way as above, the size of $\boldsymbol{U}$ can be reduced by a factor of $n_u/n'_u = s$. Applying both modifications, the number of group elements in `params` can be reduced to $n_u/s + n_m/t + 5$.

6.2 Security of Our Modified Scheme

The security analysis of our scheme, when using the above idea, is very similar to the analysis presented in Section 5. However, a few modifications to the construction of F, J, K, and L are required to ensure that these functions continue to have the properties needed in Section 5. We will assume that the same setup as in Section 5 is given and only focus on the changes needed to make the security analysis valid for our modified scheme. As in the above, we will assume that identities and messages consist of n'_u s-bit and n'_m t-bit integers respectively.

The first change is that the ranges within which the values k_u and k_m are chosen in the setup stage of the simulation, are expanded to $0 \leq k_u \leq 2^{s-1}n'_u$ and $0 \leq k_m \leq 2^{t-1}n'_m$. All other chosen values and assignments are the same as in the original setup stage of the simulation. We will assume that $l_u(2^{s-1}n'_u+1) < p$ and $l_m(2^{t-1}n'_m + 1) < p$. The functions F, J, K and L are then redefined as

$$F(\mathfrak{u}) = x' + \sum_{i=1}^{n'_u} \mathfrak{u}[i]x_i - l_u k_u \quad \text{and} \quad J(\mathfrak{u}) = y' + \sum_{i=1}^{n'_u} \mathfrak{u}[i]y_i,$$

$$K(\mathfrak{m}) = z' + \sum_{j=1}^{n'_m} \mathfrak{m}[j]z_j - l_m k_m \quad \text{and} \quad L(\mathfrak{m}) = w' + \sum_{j=1}^{n'_m} \mathfrak{m}[j]w_j$$

where $\mathfrak{u}[i]$ and $\mathfrak{m}[j]$ denote the s- and t-bit integers making up $\mathfrak{u}$ and $\mathfrak{m}$ respectively. It is easy to see that these modifications ensure that the following hold:

- $u' \prod_{i=1}^{n'_u} u_i^{\mathfrak{u}[i]} = g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})}$ and $m' \prod_{j=1}^{n'_m} m_j^{\mathfrak{m}[j]} = g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})}$
- $F(\mathfrak{u}) = 0 \mod p$ implies $F(\mathfrak{u}) = 0 \mod l_u$ and $K(\mathfrak{m}) = 0 \mod p$ implies $K(\mathfrak{m}) = 0 \mod l_m$
- If $F(\mathfrak{u}) = 0 \mod l_u$ then there is a unique choice of k_u with $0 \leq k_u \leq 2^{s-1}n'_u$ such that $F(\mathfrak{u}) = 0 \mod p$. Similarly, if $K(\mathfrak{m}) = 0 \mod l_m$ then there is a unique choice of k_m with $0 \leq k_m \leq 2^{t-1}n'_m$ such that $K(\mathfrak{m}) = 0 \mod p$.

With these properties, the other stages of the simulation for the modified scheme can be carried out just as described in the original simulation in Section 5.

The analysis of the success probability of the simulation is almost identical to the analysis in Section 5, since only the increased range of the values k_u and k_m (i.e. the last property listed above) affects the treated probabilities. This changes the probability of the events A^* and B^* when defined with the modified F and K functions and we get

$$\Pr[A^*] = \frac{1}{l_u(2^{s-1}n'_u + 1)} \quad \text{and} \quad \Pr[B^*] = \frac{1}{l_m(2^{t-1}n'_m + 1)}.$$

Since the probabilities of the events $(\bigwedge_{i=1}^{q_I} A_i | A^*)$ and $(\bigwedge_{j=1}^{q_M} B_j | B^*)$ do not change with the modifications to F and K, the success probability of the simulation is

$$\Pr[\neg\texttt{abort}] \geq \frac{1}{16(q_e + q_s)q_s(2^{s-1}n'_u + 1)(2^{t-1}n'_m + 1)}.$$

This is approximately a factor of $2^{s-1}2^{t-1}/(st)$ lower than the success probability of the simulation in Section 5.

The time complexity of the simulation remain as $t + O\big((q_e n'_u + q_s(n'_u + n'_m))\rho + (q_e + q_s)\tau\big)$ where t is the time taken by the adversary, ρ is the time for a multiplication in $\mathbb{G}$ and τ is the time for an exponentiation in $\mathbb{G}$.

6.3 Tradeoffs Between Size, Computation and Security

The above result means that we can reduce the number of elements in the public parameters to $n_u/s + n_m/t + 5$, but this will be at the cost of a loss in security of $s + t - 2 - \log_2(st)$ bits compared to the original scheme. For small values of s and t, it may be acceptable simply to trade the loss of security for the increased efficiency, which is the approach suggested by Naccache.

However, it is possible to avoid the loss of security by increasing the computational cost of the scheme. The idea, which was suggested by Chatterjee-Sarkar, is to increase the size of $\mathbb{G}$ to increase the security level provided by the CDH problem in $\mathbb{G}$ to compensate for the loss of security in the security proof caused by a given choice of s and t values. By choosing $|\mathbb{G}|$ carefully, it is possible to maintain a given security level for any choice of s and t. However, there are several factors to take into account when this approach is taken.

First of all, the level of security provided by the CDH problem in $\mathbb{G}$ will need to be estimated. Currently, the best known way of solving the CDH problem is by solving the discrete logarithm problem (DLP). The DLP in $\mathbb{G}$ can easily be reduced to the DLP in $\mathbb{G}_T$ and hence, the best known algorithms for solving the DLP in both $\mathbb{G}$ and $\mathbb{G}_T$ will need to be considered when choosing these groups. Secondly, the availability of a suitable class of elliptic curves that enables the construction of a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with $\mathbb{G}$ and $\mathbb{G}_T$ satisfying the above requirements, will need to be considered. We note that the authors of [SC05] do not consider this issue in their analysis. Finally, the increase in size of $\mathbb{G}$ and $\mathbb{G}_T$ will lead to larger space requirements for a single group element and will increase the complexity of the arithmetic in these groups.

All of these issues are important for evaluating the efficiency and security of the scheme when the size of the public parameters is reduced. However, we do not include the detailed analysis here.

7 Scheme Construction Using General Curves

Currently, the only known way of constructing a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is by using a Weil or Tate pairing on an elliptic curve. Furthermore, if $\mathbb{G}_1 = \mathbb{G}_2$, as we assume in our construction, we will be limited to using supersingular elliptic curves, a very limited class of curves. However, our scheme can easily be generalized to work with a bilinear map of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, allowing the use of wider classes of elliptic curves [Gal05]. This flexibility is important for implementation and for the selection of parameters meeting a particular concrete security level.

For our security proof to work for the generalised scheme, we require that an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ exists and that ψ maps the generator of $\mathbb{G}_2$ to the generator of $\mathbb{G}_1$. This isomorphism will only be needed in the security proof and will not be a part of the scheme itself. Furthermore, the security of the scheme will be reduced to the co-Diffie-Hellman problem on $(\mathbb{G}_1, \mathbb{G}_2)$ [BLS04]. Full details can be found in [PS02].

8 Conclusion

We have presented the first direct construction of an identity-based signature scheme that is provably secure in the standard model. Our basic scheme is computationally efficient, and we have presented a variety of techniques to improve its space requirements and to increase the range of parameter choices. It is easy to see that our construction can be generalised to produce a hierarchical IBS (HIBS) scheme. However, it is still an open problem to construct an efficient HIBS scheme that is secure in the standard model and has a tight security reduction.

Acknowledgment

We thank Jonathan Katz, Eike Kiltz and an anonymous reviewer for pointing out the folklore certification-based construction for identity-based signatures.

References

[ADR02] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *LNCS*, pages 83–107. Springer, 2002.

[Bar] P. S. L. M. Barreto. The pairing-based crypto lounge. `http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html`.

[BB04a] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In Cachin and Camenisch [CC04], pages 223–238.

[BB04b] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, 2004.

[BB04c] D. Boneh and X. Boyen. Short signatures without random oracles. In Cachin and Camenisch [CC04], pages 56–73.

[BBP04] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Cachin and Camenisch [CC04], pages 171–188.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BLMQ05] P. S. L. M. Barreto, B. Libert, N. McCullagh, and J. Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In B. Roy, editor, *ASIACRYPT*, volume 3788 of *LNCS*, pages 515–532. Springer, 2005.

[BLS04] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[BNN04] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In Cachin and Camenisch [CC04], pages 268–286.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of CCS 1993*, pages 62–73. ACM Press 1993.

[CC03] J. C. Cha and J. H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In Desmedt [Des02], pages 18–30.

[CC04] C. Cachin and J. Camenisch, editors. *Proc. of EUROCRYPT 2004*, volume 3027 of *LNCS*. Springer, 2004.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *STOC*, pages 209–218, 1998.

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *IMA Int. Conf.*, volume 2260 of *LNCS*, pages 360–363. Springer, 2001.

[Des02] Y. Desmedt, editor. *Proc. of PKC 2003*, volume 2567 of *LNCS*. Springer, 2003.

[DKXY03] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In Desmedt [Des02], pages 130–144.

[Gal05] S. D. Galbraith. Pairings. In G. Seroussi I.F. Blake and N.P. Smart, editors, *Advances in Elliptic Curve Cryptography*, pages 183–212. Cambridge University Press, 2005.

[GMR88] S. Goldwasser, S. Micali, and R.L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GS02] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.

[Hes02] F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. M. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *LNCS*, pages 310–324. Springer, 2002.

[KMPR05] E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan. Append-only signatures. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *ICALP*, volume 3580 of *LNCS*, pages 434–445. Springer, 2005.

[Nac05] D. Naccache. Secure and *practical* identity-based encryption. Cryptology ePrint Archive, Report 2005/369, 2005. `http://eprint.iacr.org/`.

[Pat02] K. G. Paterson. ID-based signatures from pairings on elliptic curves. *IEE Electronics Letters*, 38(18):1025–1026, 2002.

[PS02] K. G. Paterson and J. C. N. Schuldt. Efficient identity-based signatures secure in the standard model. Cryptology ePrint Archive, Report 2006/080, 2006. `http://eprint.iacr.org/`.

[SC05] S. Chatterjee and P. Sarkar. Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. *Proceedings of ICISC*, 2005. To appear.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 84*, pages 47–53, 1984.

[Wat05] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

[Yi03] X. Yi. An identity-based signature scheme from the Weil pairing. *IEEE Communications Letters*, 7(2), 2003.

Event-Oriented k-Times Revocable-iff-Linked Group Signatures

Man Ho Au[1], Willy Susilo[1], and Siu-Ming Yiu[2]

[1] Center for Information Security Research
School of Information Technology and Computer Science
University of Wollongong, Wollongong 2522, Australia
{mhaa456, wsusilo}@uow.edu.au
[2] Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong
smyiu@cs.hku.hk

Abstract. In this paper, we introduce the notion of event-oriented k-times revocable if and only if linked group signatures (k-EoRiffL group signatures). In k-EoRiffL group signatures, signers can sign on behalf of a group anonymously and unlinkably up to a permitted number of times (k) per *event*. No party, even the group manager, can revoke the anonymity of the signer. On the other hand, everyone can identify the signer if he signs more than k times for a particular *event*. We then show that k-EoRiffL group signatures can be used for k-times anonymous authentication(k-TAA), compact e-cash, e-voting, etc.

We formally define security model for the new notion and propose constant-size construction, that is, size of our construction is *independent* of the size of the group and the number of permitted usage k. Our construction is secure based on the q-strong Diffie-Hellman assumption and the y-DDHI assumption.

Keywords: event-oriented, revocable anonymity, group signature, k-TAA.

1 Introduction

In the age of information technology, number of applications over the Internet continues to grow. These include messaging, voting, payments, commerce, etc. At the same time, people are concerned with their personal privacy and are aware of the protection of privacy.

Anonymity is an important form of privacy protection. This is especially true in case of group-oriented cryptography, where a group of users are involved. In schemes where participation of one or a proper subset of members is required to complete a process, anonymity refers to whether participants are distinguishable from non-participants. Users may prefer perfect anonymity, meaning that it is not possible to distinguish participants from non-participants so as to maintains their privacy in participating the process. In [3], anonymity can be divided

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 223–234, 2006.
© Springer-Verlag Berlin Heidelberg 2006

into 4 different levels, namely, *No Anonymity*, *Revocable Anonymity*, *Linkable Anonymity* and *Full Anonymity* accordingly. Extending their ideas, we further refine levels of anonymity for group-oriented cryptography as follow, from highest level to lowest level (no anonymity).

1.1 Levels of Anonymity for Group–Oriented Cryptography

Full Anonymity. It means that identity of the participating user is indistinguishable from the non-participating users by *any* party. A prominent example is ring signature, formalized in [19]. Many ring signatures are then proposed subsequently and the constant-size construction (meaning the size of the signature is independent of the size of the group) first appeared in [10], followed by [16].

Linkable Anonymity. Users can participate in the process anonymously but their participation are linked, that is, everybody can tell if the underlying participant in two separate processes are the same. An example is linkable ring signature[13, 25, 24], where everybody can tell if two signatures are generated from the same signer. However, no one can tell who the actual signer is. A generalized notion is k-times linkable anonymity, meaning that suppose the user participate for k times or less, he enjoys full anonymity while if he participate for more than k times, at least two of his participations are linked.

Revocable-iff-linked Anonymity. Similar to linkable anonymity, users enjoy full anonymity if they only participate once. However, if they participate twice, everybody can reveal their identity. Some e-cash scheme [7, 2], tracing-by-linking (TbL) group signature scheme[26] are examples of this type. In [7, 2], no one (even the bank) could revoke the anonymity of the spender of the e-cash while in case someone spends twice, his identity is revealed. A more general notion is k-times Revocable-iff-Linked anonymity, in which user identity is revealed if he participate for more than k times. Examples include compact e-cash scheme[8], k-times anonymous identification (k-TAA)[21, 17].

Revocable Anonymity. Basically it means anonymity to everybody except an *Open Authority(OA)*. From user's standpoint, it can be regarded as a lower anonymity level than Revocable-iff-Linked anonymity since in the user must trust the OA not to abuse his power in comparison with Revocable-iff-Linked anonymity where users are anonymous unless they break the condition themselves. Group signature[1] is a famous example.

Linkable and Revocable Anonymity. As its name suggest, users enjoy linkable anonymity towards everybody except OA, where OA can always revoke the anonymity of the user. Systems where users are identified by pseudonym[12] with an authority knowing the corresponding identity of the user for each pseudonym belongs to this category. Many e-cash schemes[9, 23] in fact belongs to this category too. Should a user double-spends, everybody can detect it and the OA can then reveal the identity of the cheater.

Revocable-iff-Linked and Revocable Anonymity. Similarly, users enjoy revocable-iff-linked anonymity to everybody except OA. In fact, Linkable (resp.

Anonymity Level	Examples	Size	Event-Oriented	Ad-hoc
Full	Ring Sign[19]	O(n)	N/A	✓
	Anon Ident[10, 16]	O(1)	N/A	✓
Linkable	Linkable Ring[13]	O(n)	×	✓
	Eo-Linkable Ring[25]	O(n)	✓	✓
Revocable-iff-Linked				
2-times	E-Cash[7, 2],TbL[26]	O(1)	×	×
k-times	Compact E-Cash[8]	O(1)	×	×
	k-TAA[21]	O(k)	✓	×
	dynamic k-TAA[17]	O(k)	✓	✓
	constant-size K-TAA[22]	O(1)	✓	×
	this paper	O(1)	✓	×
Full+OA	Group Signatures	O(1)	×	×
Link+OA	Fair E-Cash[9, 23]	O(1)	×	×

Fig. 1. Examples of group-oriented cryptographic schemes with different levels of anonymity

Revocable-iff-Linked) and Revocable Anonymity can be achieved by adding an identity escrow to the schemes with linkable anonymity (resp. Revocable-iff-Linked anonymity).

No Anonymity. Identity-based signature[20] is an example of group-oriented cryptography with no anonymity. Multi-signatures[14] is another example if we assume that each user is in possession of one public key only.

As stated in [3], our goal is to decide schemes with carefully adjusted level of anonymity suitable for the application. For example, ring signature is perfect for secret leaking. In an e-voting scheme, linkable anonymity or revocable-iff-linked anonymity is essential for detection of double-vote. In e-voting, linkable anonymity may be acceptable since in the vote-counting stage, the party can disregard those who double-vote. People who double-vote thus would not gain any real benefit. On the other hand, in e-cash scheme, double-spender must be caught and thus revocable-iff-linked anonymity is a must. A work around is to use scheme with linkable and revocable anonymity so that when double-spender is caught, the OA could find out who the cheater is. The problem of this work around is that anonymity of honest spender is no longer assured and trust is placed on OA not to abuse its power.

1.2 Concept of Event in Linkable Anonymity

The concept of event-oriented linkability is introduced in [25]. *Event-oriented* linkable group/ring signatures means that one can tell if two signatures are linked if and only if they are signed for the same event, despite the fact that they may be signed on behalf of different groups. This considerably add flexibility to schemes with linkable (resp. revocable-iff-linked) anonymity since user needs not obtain new secret key for different events.

In group-oriented cryptography, other concerns include whether the group can be formed in an ad-hoc manner or users must register with some group

manager first. Ring signature and group signature are example of each type respectively. Order of computation and space complexity are other concerns. Figure 1.2 categorizes *some* of the schemes in existing literature according to their level of anonymity.

1.3 Related Works

Very recently and independently, Teranishi and Sako [22] proposed an k-TAA scheme with constant proving cost. Their construction is very similar to ours and is of similar performance. Our scheme can be thought of as the non-interactive version of theirs.

Our Contributions. We introduce a new notion, event-oriented k-times revocable-iff-linked group signatures, which belongs to the Revocable-iff-Linked Anonymity category. With the event-oriented feature, this new notion is flexible for many applications such as compact e-cash, e-voting, k-times anonymous identification, to name a few. Our notion is closely related to k-TAA if we treat each content provider in k-TAA as event. Specifically, we make the following contributions

- We introduce the notion of event-oriented k-times revocable-iff-linked group signatures.
- We propose constant-size construction.
- We show how to turn k-EoRiffL group signatures into compact e-cash and k-TAA. Our scheme can be used to construct k-TAA whose size is independent of the group and also independent of k.
- We formalize the security model for k-EoRiffL group signatures and present security arguments for our scheme.

Organization. We discuss related works and technical preliminary in the next section. Security model is shown in section 3. The construction of k-EoRiffL Group Signatures is shown in section 4, accompanied by security analysis. Finally we conclude the paper with applications and some discussions in section 5.

2 Preliminaries

2.1 Notations

Let $\hat{e}$ be a bilinear map such that $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic multiplicative groups of prime order p.
- each element of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ has unique binary representation.
- g_0, h_0 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
- $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(h_0) = g_0$.
- (Bilinear) $\forall x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and a, $b \in \mathbb{Z}_p$, $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$.
- (Non-degenerate)$\hat{e}(g_0, h_0) \neq 1$.

$\mathbb{G}_1$ and $\mathbb{G}_2$ can be same or different groups. We say that two groups ($\mathbb{G}_1$, $\mathbb{G}_2$) are a bilinear group pair if the group action in $\mathbb{G}_1$, $\mathbb{G}_2$, the isomorphism ψ and the bilinear mapping $\hat{e}$ are all efficiently computable.

2.2 Mathematical Assumptions

Definition 1 (Decisional Diffie-Hellman). *The Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}$ is defined as follow: On input a quadruple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, output 1 if $c = ab$ and 0 otherwise. We say that the (t, ϵ)-DDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ over random guessing in solving the DDH problem in $\mathbb{G}$.*

Definition 2 (q-Strong Diffie-Hellman[5]). *The q-Strong Diffie-Hellman (q-SDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follow: On input a $(q+2)$-tuple $(g_0, h_0, h_0^x, h_0^{x^2}, \cdots, h_0^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, output a pair (A, c) such that $A^{(x+c)} = g_0$ where $c \in \mathbb{Z}_p^*$. We say that the (q, t, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.*

Definition 3. y-Decisional Diffie-Hellman Inversion Assumption[11, 8]. *The y-Decisional Diffie-Hellman Inversion problem (y-DDHI) in prime order group $\mathbb{G}$ is defined as follow: On input a $(y+2)$-tuple $g, g^x, g^{x^2}, \cdots, g^{x^y}, g^c \in \mathbb{G}^{y+2}$, output 1 if $c = 1/x$ and 0 otherwise. We say that the (y, t, ϵ)-DDHI assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ over random guessing in solving the y-DDHI problem in $\mathbb{G}$.*

2.3 Building Blocks

Verifiable Random Function. One of the building blocks of our k-ERiffL group signatures is the verifiable random function (VRF) from [11]. The notion VRF was introduced by Micali, Rabin and Vadhan in [15]. Roughly speaking, an VRF is a pseudo-random function with non-interactive proof of correctness of its output. The VRF defined in [11] is described as follow. The function f is defined by a tuple $(\mathbb{G}_p, p, g, s)$, where $\mathbb{G}_T$ is a cyclic group of prime order p, g a generator of $\mathbb{G}_p$ and s is a seed in $\mathbb{Z}_p$. On input x, $f_{\mathbb{G}_p,p,g,s}(x) = g^{\frac{1}{s+x+1}}$. Efficient proof such that the output is correctly formed (with respect to s and x in some commitment scheme such as Pedersen Commitment [18]) exists and the output of f is indistinguishable from random elements in $\mathbb{G}_p$ if the y-DDHI assumption in $\mathbb{G}_p$ holds.

3 Security Model

3.1 Syntax

An event-oriented k-times revocalbe-iff-linked group signature is a tuple (GMSetup, UserSetup, Join, Sign, Verify, Link, Revoke) of seven polynomial time algorithms. The following enumerates the syntax.

- GMSetup On input an unary string 1^λ, where λ is a security parameter, the algorithm outputs GM secret key gsk and group public key gpk. All algorithms below have implicitly gpk as one of their inputs.

- UserSetup On input 1^λ, randomly outputs a key pair (pk, sk).
- Join Protocol. User with input (pk,sk) engage with GM with input (gsk). Finally the user obtain a *cert* which allow it to sign on behalf of the group.
- Sign User with input message $m \in \{0,1\}^*$, an event identifier $evt \in \{0,1\}^*$, pk, sk, *cert* output a signature σ.
- Verify Verifier with input message $m \in \{0,1\}^*$, event identifier $evt \in \{0,1\}^*$, signature σ output accept or reject.
- Link On input two signatures σ_1, σ_2, output link or unlink.
- Revoke On input two signatures σ_1, σ_2 such that link $\leftarrow$ Link(σ_1, σ_2), output pk^*.

A event-oriented k-times revocable-iff-linked group signature must satisfy

1. *Verification Correctness.* Signatures signed according to specification are accepted during verification, with overwhelming probability;
2. *Linking Correctness.* If two signatures are linked, they must be generated from the same signer. In addition, the output of Revoke of this two signature must be the actual signer if they two signatures are on different messages.

3.2 Security Notions

We first gives an informal description of the security requirement. A *secure* k-EoRiffL Group Signatures scheme should possess *linkability*, *anonymity* and *non-slanderability*, introduce as follows.

- *Linkability.* Roughly speaking, linkability means that a user cannot sign, per event, more than the allowable times without being linked. More precisely, we required that collusion of n users cannot produce more than nk valid signatures or in case they do produce $nk+1$ signature for a particular event, at least one of the colluder must be identified. A related notion is revocability, which means that from the linked signatures, identity of the actual signer must be revealed. It is straight forward to see that revocability is implied by the definition of linkability.
- *Anonymity.* It is required that no collusion of users and GM can ever guess who the actual signer is in a group signature with probability better than random guessing.
- *Non-slanderability.* It is required that an honest user cannot be accused of having sign more than k times, even with the help of GM.

In revocable-iff-linked group signatures, the standard notion of *unforgeability* is implied by *linkability* and *non-slanderability*. For if someone can forge a signature, either he can generate the signature without being *linked* or he successfully *slander* an honest user.

The capability of an adversary $\mathcal{A}$ is modeled as oracles.

- Join Oracle: $\mathcal{A}$ present a public key pk and engages in the join protocol as user and obtains a certificate. The oracle stores pk in a set $\mathbb{X}_A$.

- Signing Oracle: On input a message m, and event evt, the oracle return a signature σ on m and evt.
- Hash Oracle: $\mathcal{A}$ can ask for the values of the hash functions for any input.

We require that the answers from the oracles are indistinguishable from the view as perceived by an adversary in real world attack.

Definition 4 (Game Linkability)

- (Initialization Phase.) *The challenger $\mathcal{C}$ takes a sufficiently large security parameter λ and runs* `GMSetup` *to generate gpk and also a master secret key gsk. $\mathcal{C}$ keeps gsk to itself and sends gpk to $\mathcal{A}$.*
- (Probing Phase.) *The adversary $\mathcal{A}$ can perform a polynomially bounded number of queries to the oracles in an adaptive manner.*
- (End Game Phase.) *Let q_j be the number of queries to the Join Oracle. $\mathcal{A}$ submits an event evt∗, signatures σ_i on message m_i and evt^*, $i = 1, \cdots, kq_j + 1$ to $\mathcal{C}$.*

$\mathcal{A}$ wins the game if all the following holds:

1. *all σ_i are valid*
2. *none of the σ_i are the output of the Signing Oracle*
3. *None of the σ_i are linked or they are linked but* Revoke *cannot pointed to any of the users during the join protocol query.*
4. *$m_i \neq m_j$ if $i \neq j$.*

The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

In the above game, if the condition such that each m_i are different is replaced by each σ_i are different, then we refer to the game as Game Strong Linkability.

Definition 5 (Game Anonymity)

- (Initialization Phase.) *The challenger $\mathcal{C}$ takes a sufficiently large security parameter λ and runs* `GMSetup` *to generate gpk and also the master secret key gsk. $\mathcal{C}$ gives* both *gpk and gsk to the user. Since $\mathcal{A}$ is in possession of gsk, only Hash oracle query is allowed in Game Anonymity.*
- (Challenge Phase.) *$\mathcal{C}$ runs the* Join *protocol with $\mathcal{A}$ acting as GM to obtain a certificate $cert_0$. $\mathcal{C}$ generate another certificate $cert_1$ by himself. $\mathcal{A}$ is then allowed to issue the following special signature query by submitting event evt_i, message m_i, bit $b_i = 0$ or 1 for the i-th special signature query. $\mathcal{C}$ return a signature on evt_i, m_i using $cert_{b_i}$. The only restriction is that for a particular event, the number of signature query for $cert_0$ or $cert_1$ does not exceed k. Finally, $\mathcal{A}$ gives evt^*, m^* to $\mathcal{C}$, $\mathcal{C}$ uses $cert_b$, where $b \in \{0, 1\}$ is the output of a fair coin, to sign on evt^*, m^* and return the signature to $\mathcal{A}$.*
- (End Game Phase.) *The adversary $\mathcal{A}$ decides $b = 0$ or 1.*

$\mathcal{A}$ wins the above game if it guesses correctly. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins minus $\frac{1}{2}$.

Definition 6 (Game Non-Slanderability)

- (Initialization Phase.) *The challenger $\mathcal{C}$ takes a sufficiently large security parameter λ and runs* `GMSetup` *to generate gpk and also the master secret key gsk. $\mathcal{C}$ gives* both *gpk and gsk to the user. Since $\mathcal{A}$ is in possession of gsk, only Hash oracle query is allowed in Game Non-Slanderabiliy.*
- (Challenge Phase.) *$\mathcal{C}$ runs the* Join *protocol for q_j times with $\mathcal{A}$ acting as GM to obtain a certificate $cert_j$. $\mathcal{A}$ is then allowed to issue the following special signature query by submitting event evt_i, message m_i, $b_i = 1, \cdots, q_j$ for the i-th special signature query. $\mathcal{C}$ return a signature on evt_i, m_i using $cert_{b_i}$. The only restriction is that for a particular event, the number of signature query for any of the cert does not exceed k. $\mathcal{A}$ is also allowed to corrupt the user corresponding to $cert_j$.*
- (End Game Phase.) *$\mathcal{A}$ submits an event $evt*$, two signatures σ_0^*, σ_1^* on message m_0, m_1 and evt^*. $\mathcal{A}$ wins the game if σ_0^*, σ_1^* is linked and* Revoke *on the two linked signature is a user in one of the join query and is not corrupted.*

The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

A k-EoRiffL Group signature is *secure* if no PPT adversary can win in Game Linkability, Game Anonymity and Game Non-Slanderability with non-negligible advantage. It is *strongly secure* if it is secure and no PPT adversary can win in Game Strong Linkability.

4 Our Construction

4.1 Our *k*-EoRiffL Group Signature

GMSetup. Let λ be the security parameter. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair with computable isomorphism ψ as discussed such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime p of λ bits. Also assume $\mathbb{G}_p$ be a group of order p where DDH is intractable. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H_{evt}\{0,1\}^* \rightarrow \mathbb{G}_p$ be cryptographic hash functions. Let g_0, g_1, g_2, g_3 be generators of $\mathbb{G}_1$, h_0, h_1, h_2, h_3 be generators of group $\mathbb{G}_2$ such that $\psi(h_i) = g_i$ and u_0, u_1, u_2, u_3 be generators of $\mathbb{G}_p$ such that relative discrete logarithm of the generators are unknown. This can be done by setting the generators to be output of a hash function of some publicly known seed. The group manager (GM) also randomly selects $\gamma \in_R \mathbb{Z}_p^*$ and compute $w = {h_0}^{\gamma}$. The group public key is $gpk = (g_0, g_1, g_2, g_3, h_0, w, u_0, u_1, u_2, u_3, k)$ and the GM secret key is $gsk = \gamma$. k has to be much smaller than 2^{λ}.

UserSetup. We assume PKI, that is, each user is equipped with a discrete logarithm type public and private key pairs $({u_0}^x, x) \in \mathbb{G}_p \times \mathbb{Z}_p^*$.

Join Protocol. **Idea:** User with public key $y = {u_0}^x$ joins the group by obtaining a cert in the form of (A, e) such that $A^{e+\gamma} = g_0 g_1^s g_2^t g_3^x$ for some random number s, t unknown to GM.

Actual Protocol

1. User randomly selects $s' \in_R \mathbb{Z}_p^*$ and sends $C' = g_1^{s'} g_2^t g_3^x$, along with the proof $\Pi_0 = SPK\{(s', t, x) : C' = g_1^{s'} g_2^t g_3^x \wedge y = u_0^x\}$ to GM.
2. GM verifies that Π_0 is valid and randomly selects $s'' \in_R \mathbb{Z}_p^*$. It computes $C = C' g_1^{s''}$ and selects $e \in_R \mathbb{Z}_p^*$. It then computes $A = (g_0 C)^{\frac{1}{e+\gamma}}$ and sends (A, e, s'') to the user.
3. User computes $s = s' + s''$and checks if $\hat{e}(A, w h_0^e) = \hat{e}(g_0 g_1^s g_2^t g_3^x, h_0)$. It then stores (A, e, s, t).

Sign. **Idea:** For each event $evt \in \{0,1\}^*$, the user manages a counter J_{evt} which is the number of signatures he has generated. When the counter reaches k, user can no longer signs anonymously.

For a particular event evt^* and message m such that $R = H(evt^*, m)$ and $u_{evt^*} = H_{evt}(evt^*)$, user with (A, e, s, t) from GM and counter $J_{evt^*} \leq k$ produces a signature of knowledge by submitting $S = u_{evt^*}^{\frac{1}{J_{evt^*}+s+1}}, T = u_0^x u_{evt^*}^{\frac{R}{J_{evt^*}+t+1}}$ and proves, in zero-knowledge manner, (1) - (4).

1. $A^{e+\gamma} = g_0 g_1^s g_2^t g_3^x$ (This shows that the user possess a certificate from GM.)
2. $S = u_{evt^*}^{\frac{1}{J_{evt^*}+s+1}}$. ($S$ is called a *linkability tag* or simply *tag*. For each certificate (A, e, s, t) and event evt^*, user can generate k valid *tag*. Suppose a user generate more than k *tags* from the same certificate, duplicate *tags* must be used and can thus be detected.)
3. $0 \leq J_{evt^*} \leq k$ (The number of signings does not exceed k.)
4. $T = u_0^x u_{evt^*}^{\frac{R}{J_{evt^*}+t+1}}$ (Component for revealing identity of user using duplicated tags.)

Should a user attempt to sign more than the permitted number of times k for a particular event, he must have used duplicated *tag* and can thus be detected. Then two transcripts with the same *tag* together with different T reveals identity of user. Details are shown in the revoke algorithm. On the other hand, anonymity of honest signer is guaranteed. In short, the above can be represented by

$$\Pi_1 = SPK\{(A, e, s, t, x, J_{evt^*}) : A^{e+\gamma} = g_0 g_1^s g_2^t g_3^x \wedge S = u_{evt^*}^{\frac{1}{J_{evt^*}+s+1}}$$
$$\wedge T = u_0^x u_{evt^*}^{\frac{R}{J_{evt^*}+t+1}} \wedge 0 \leq J_{evt^*} \leq k\}(M)$$

Remarks: Two signature for the same event can be falsely 'linked' if $J_{evt^*} + s = J'_{evt^*} + s'$. However, the probability is negligible if $k << 2^\lambda$.

Details of Sign (instantiation of SPK Π_1) is shown in Appendix A.

Verify. The verifier verifies the SPK.

Link. For the same *event* evt^*, two signatures are from the same user (*linked*) if they share the same *tag* S.

Revoke. Given two signatures with same *tag* for the same event evt^* and different messages m, anyone can compute $u_o^x = (\frac{T^{R'}}{T'^R})^{((R'-R)^{-1})}$.

4.2 Security Analysis

Regarding our construction, we have the following theorem, whose proof can be found in the full version of the paper [4].

Theorem 1. *Our k-RiffL group signature is secure under the q-SDH assumption and the y-DDHI assumption in the random oracle model.*

We remark that our scheme does not possess strong linkability, meaning that two linked signatures of the same message from the same signer is not revocable. Thus, our scheme maybe best suit to be used in interactive protocols where (part of) the message maybe provided by another party to ensure its uniqueness. Examples include transaction information provided by the merchant in e-cash or a random seed provided by the content provider in k-TAA.

5 Applications and Discussions

5.1 Constant-Size k-TAA

If the verifier's identity is appended to the event *evt*, then we have a event-oriented k-times revocable-iff-linked group signatures which is verifier-specific, that is, the signatures for different verifiers will not be linked with one another. It is straight forward to show that k-TAA can be constructed from k-EoRiffL group signature by setting the event to be the identity of the verifier(content provider). Our k-EoRiffL group signature is the first k-TAA scheme which is of size O(1) (independent of group size and k).

On the other hand, if we treat the tags published by the content provider in k-TAA as event then any k-TAA can be turned into k-EoRiffL group signatures if the underlying k-TAA can be done in an non-interactive manner.

5.2 Compact E-Cash

In fact, compact E-Cash can be viewed as a k-times revocable-iff-linked group signature. To use the k-RiffL group signature as a compact e-cash scheme, the bank plays the role of GM in the scheme while the join protocol can be treated as users obtain a wallet from the bank (i.e. withdrawing k coins). Spending is done by using the certificate to sign. Since each certificate can be used to sign k-times, each wallet possesses k coins. When the wallet is used up, user need to obtain another certificate from the bank. Note that the concept of event is not used.

References

1. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, pages 255–270, 2000.
2. Man Ho Au, Sherman S. M. Chow, and Willy Susilo. Short e-cash. In *INDOCRYPT*, pages 332–346, 2005.

3. Man Ho Au, Joseph K. Liu, Patrick P. Tsang, and Duncan S. Wong. A Suite of ID-Based Threshold Ring Signature Schemes with Different Levels of Anonymity. Cryptology ePrint Archive, Report 2005/326, 2005. `http://eprint.iacr.org/`.
4. Man Ho Au, Willy Susilo, and Siu-Ming Yiu. Event-Oriented k-times Revocable-iff-Linked Group Signatures, 2006. Full version.
5. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT*, pages 56–73, 2004.
6. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT*, pages 431–444, 2000.
7. Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *CRYPTO*, pages 302–318, 1993.
8. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *EUROCRYPT*, pages 302–321, 2005.
9. Sébastien Canard and Jacques Traoré. On fair e-cash systems based on group signature schemes. In *ACISP*, pages 237–248, 2003.
10. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *EUROCRYPT*, pages 609–626, 2004.
11. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *PKC 2005*, volume 3386 of *LNCS*, pages 416 – 431, 2005.
12. Wei-Bin Lee and Chang-Kuo Yeh. A new delegation-based authentication protocol for use in portable communication systems. *IEEE Trans. Wireless Commun.*, 4(1):57–64, January 2005.
13. Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract). In *ACISP 2004*, volume 3108 of *LNCS*, pages 325–335. Springer-Verlag, 2004.
14. Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In *ACM Conference on Computer and Communications Security*, pages 245–254, 2001.
15. Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.
16. Lan Nguyen. Accumulators from Bilinear Pairings and Applications. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292, 2005.
17. Lan Nguyen and Rei Safavi-Naini. Dynamic k-Times Anonymous Authentication. Cryptology ePrint Archive, Report 2005/168, 2005. `http://eprint.iacr.org/`.
18. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
19. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565, 2001.
20. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO 1984*, volume 196 of *LNCS*, pages 47–53, 1984.
21. Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-Times Anonymous Authentication (Extended Abstract). In *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 308–322. Springer-Verlag, 2004.
22. Isamu Teranishi and Kazue Sako. k-times anonymous authentication with a constant proving cost. In *Public Key Cryptography*, pages 525–542, 2006.
23. Mårten Trolin. A universally composable scheme for electronic cash. In *INDOCRYPT*, pages 347–360, 2005.
24. Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *ISPEC*, pages 48–60, 2005.

25. Patrick P. Tsang, Victor K. Wei, Tony K. Chan, Man Ho Au, Joseph K. Liu, and Duncan S. Wong. Separable Linkable Threshold Ring Signatures. In *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 384–398. Springer-Verlag, 2004.
26. Victor K. Wei. Tracing-by-linking group signatures. In *ISC*, pages 149–163, 2005.

A Detail of Sign Algorithm (Instantiation of Π_1)

Suppose $u_{evt^*} = H_{evt}(evt^*)$ and $J_{evt^*} < k$, $R = H(evt^*, m)$, the signer first computes the following quantities $A_1 = g_1^{r_1} g_2^{r_2}$, $A_2 = A g_2^{r_1}$, $A_3 = g_1^{J_{evt^*}} g_2^t g_3^{r_3}$, for $r_1, r_2, r_3 \in_R \mathbb{Z}_p^*$, in $\mathbb{G}_1$. Compute *tag* $S = u_{evt^*}^{\frac{1}{J_{evt^*}+s+1}}$, $T = u_0^x u_{evt^*}^{\frac{R}{J_{evt^*}+t+1}}$. The signer then computes a signature of knowledge (instantiation of Π_1) as follows.

$\Pi_2 = SPK\{(r_1, r_2, r_3, \delta_1, \delta_2, \delta_3, \delta_J, \delta_t, e, s, t, x, J_{evt^*}) : A_1 = g_1^{r_1} g_2^{r_2} \wedge A_1^e = g_1^{\delta_1} g_2^{\delta_2} \wedge \frac{e(A_2,w)}{e(g_0,h_0)} = e(g_1,h_0)^s e(g_2,h_0)^t e(g_3,h_0)^x e(g_3,h_0)^{\delta_1} e(g_2,w)^{r_1} e(A_2,h_0)^{-e} \wedge \frac{u_{evt^*}}{S} = S^{J_{evt^*}} S^s \wedge A_3 = g_1^{J_{evt^*}} g_2^t g_3^{r_3} \wedge A_3^x = g_1^{\delta_J} g_2^{\delta_t} g_3^{\delta_3} \wedge \frac{u_{evt^*}^R}{T} = T^{J_{evt^*}} T^t u_0^{-\delta_J} u_0^{-\delta_t} u_0^x \wedge 1 \leq J_{evt^*} \leq k\}(M)$ where $\delta_1 = r_1 e, \delta_2 = r_2 e, \delta_J = J_{evt^*} x, \delta_t = tx, \delta_3 = r_3 x$.

The range part $1 \leq J_{evt^*} \leq k$ require some attention. Secure and efficient exact proof of range is possible in groups of unknown order under factorization assumption [6]. Here, we make use of the fact that if we set $k = 2^t$ for some integer t, efficient range check, of order $O(logk)$, for J_{evt^*} could be achieved as follows.

Let g, h be two generators of a cyclic group $\mathbb{G}$ of order p whose relative discrete logarithm is unknown. To prove knowledge of a number J such that $0 < J \leq k$ in a commitment $C_J = g^J h^r$, let J_i be the i-th bit of J for $i = 1, \cdots t$. Compute $C_i = g^{J_i} h^{r_i}$ for some $r_i \in_R \mathbb{Z}_p^*$ for $i = 1, \cdots, t$. Compute the following SPK Π_{range}.

$\Pi_{range} = SPK\{(J, a, b, r, r_i) : C_J = g^J h^r \wedge C_J/g = g^a h^r \wedge \prod_{j=1}^t (C_j)^{2^j} = g^J h^b \wedge [C_i = h^{r_i} \vee C_i/g = h^{r_i}]_{i=1}^{i=t}\}(M)$ where $a = J - 1$, $b = \prod_{j=1}^t r_j 2^j$.

On the other hand, constant-size range proof is made possible as outlined in [22]. The GM has to publish k signatures $Sig(1), \cdots, Sig(k)$. In the proof, instead of proving $1 \leq J_{evt^*} \leq k$ (which has complexity $O(logk)$), the signer proves possession of signature on J_{evt^*} (which has complexity $O(1)$). This indirectly proves that J_{evt^*} is within the range. However, public key size of the GM is now linear in k, and user colluding with GM can be untraceable (since the malicious GM can issue several $Sig(J_{evt^*})$ for the user.

Key Replacement Attack Against a Generic Construction of Certificateless Signature*

Bessie C. Hu[1], Duncan S. Wong[1], Zhenfeng Zhang[2], and Xiaotie Deng[1]

[1] Department of Computer Science
City University of Hong Kong
Hong Kong, China
{bessiehu, duncan, deng}@cs.cityu.edu.hk
[2] State Key Laboratory of Information Security, Institute of Software
Chinese Academy of Sciences
Beijing 100080, China
zfzhang@is.iscas.ac.cn

Abstract. Certificateless cryptography involves a Key Generation Center (KGC) which issues a partial key to a user and the user also independently generates an additional public/secret key pair in such a way that the KGC who knows only the partial key but not the additional secret key is not able to do any cryptographic operation on behalf of the user; and a third party who replaces the public/secret key pair but does not know the partial key cannot do any cryptographic operation as the user either. We call this attack launched by the third party as the key replacement attack. In ACISP 2004, Yum and Lee proposed a generic construction of digital signature schemes under the framework of certificateless cryptography. In this paper, we show that their generic construction is insecure against key replacement attack. In particular, we show that the security requirements of their generic building blocks are insufficient to support some security claim stated in their paper. We then propose a modification of their scheme and show its security in a new and simplified security model. We show that our simplified definition and adversarial model not only capture all the distinct features of certificateless signature but are also more versatile when compared with all the comparable ones. We believe that the model itself is of independent interest.

1 Introduction

Certificateless cryptography, introduced by Al-Riyami and Paterson in 2003 [1], is intended to solve the key escrow issue which is inherent in identity-based cryptography [10, 5], while at the same time, eliminate the use of certificates as in

* The second author was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 9040904 (RGC Ref. No. CityU 1161/04E)), and the third author was supported by National Natural Science Foundation of China (No.60373039, 90604018).

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 235–246, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the conventional Public Key Infrastructure (PKI), which is generally considered to be costly to use and manage.

In a certificateless cryptosystem, a Key Generation Center (KGC) is involved to issue a *user partial key* to a user with respect to the user's identity ID, which is assumed to be unique in the system. The user also independently generates an additional *public/secret key pair* and performs some cryptographic operations in such a way that they can only be carried out when both the user partial key and the user secret key are known. Knowing only one of them should not be able to impersonate the user and carry out any of the cryptographic operations as the user. We summarize the two types of attacks against a certificateless cryptosystem as follows.

KGC Attack. The KGC who knows only the partial key but not the additional secret key of the user is not able to do any cryptographic operation as the user; and

Key Replacement Attack. A third party who can *replace* the user's public/secret key pair but does not know the user's partial key issued by the KGC cannot do any cryptographic operation as the user either.

In an identity-based cryptosystem [10, 5], the only secret of a user is generated by the KGC. Therefore, there is an inherent key escrow issue in such a cryptosystem. In a certificateless cryptosystem, on the other hand, the defense against KGC Attack is to solve this key escrow issue so that even if the KGC is an eavesdropper [1] who cannot replace the user's public/secret key pair, the KGC is not able to perform any cryptographic operation as the user.

In a conventional Public Key Infrastructure (PKI), certificates are used to 'bind' the public key of a user to the user's identity. Although key escrow is not an issue in PKI as the Certification Authority (CA) is not supposed to know the user's private key, PKI is commonly considered to be expensive to implement, use and maintain. However, if certificates are not used, a user's public key can easily be replaced by a third party. In a certificateless cryptosystem, the defense against Key Replacement Attack is to solve this problem. It requires that even if the public key is replaced, the third party should still not be able to perform any cryptographic operation as the user as long as the third party cannot compromise the user's partial key. Of course, if the third party colludes with the KGC, they may still be able to impersonate the user. But also notice that the CA can do similar damage in PKI when the CA is no longer honest. We believe that certificateless cryptography has nice features borrowed from both identity-based cryptography and conventional public key cryptography. In this paper, we focus on signature schemes in the setting of certificateless cryptography. From now on, we refer to these schemes as *certificateless signature schemes*.

The first certificateless signature scheme was proposed by Al-Riyami and Paterson [1]. However, it was recently found vulnerable to the key replacement attack by Huang et al. [8]. In [11], Yum and Lee proposed a generic construction of certificateless signature. The generic construction is built upon two primitives: a conventional digital signature scheme and an identity-based signature scheme. On the security of the generic construction, the authors claimed that the generic construction is secure against KGC attack and key replacement attack if the digital signature scheme is existential unforgeable against chosen message

attack (euf-cma) as defined in [7] and the identity-based signature scheme is existential unforgeable against chosen message and identity attack (euf-cma-ida) under a model similar to the one defined in [2]. However, we find that the security requirements above are insufficient to support their security claim. Below are the main results of our paper.

Our Results. We show that the generic construction of certificateless signature proposed by Yum and Lee in [11] is insecure against key replacement attack if the underlying euf-cma secure digital signature scheme is vulnerable to an attack called "message-and-key-replacement attack". We show by examples that some specific euf-cma secure signature schemes are vulnerable to this "message-and-key-replacement attack" (note that a digital signature scheme, which is insecure against the "message-and-key-replacement attack" can still be euf-cma secure). In our key replacement attack against the generic construction of [11], a third party who does not know a user's partial key can existentially forge a message-signature pair by replacing the user's public key. In addition, the message of the forgery can be chosen arbitrarily by the third party. This shows that their security claim ([11, Theorem 1]) cannot be correct.

We then propose a modification of their scheme and show its security in a new and simplified security model. The simplified definition and adversarial model not only capture all the distinct features of certificateless signature but are also more versatile when compared with all the comparable ones [1, 11, 8]. In particular, all the comparable ones have seven algorithms defined for a certificateless signature scheme. We show that five are enough and actually are more versatile to include new constructions which are not possible to be captured by the old definition. The security model we propose in this paper is also simplified when compared with previously proposed models. In particular, our model reduces the number of restrictions for the adversaries to win the games, and also captures some new attacks that cannot be captured in old models. We believe that the model itself is of independent interest.

Paper Organization. In Sec. 2, a new and simplified definition and adversarial model for certificateless signature are proposed. In Sec. 3, the generic construction of [11] is reviewed and shown to be insecure against key replacement attack. "Message-and-key-replacement-attack" on some concrete signature schemes are also described. In Sec. 4, an improved generic construction is proposed.

2 Security Model

A *certificateless signature scheme* is a tuple of polynomial-time algorithms denoted by (MasterKeyGen, PartialKeyGen, UserKeyGen, CL-Sign, CL-Ver). The first four algorithms may be randomized but the last one is not.

1. MasterKeyGen (Master Key Generation): On input 1^k where $k \in \mathbb{N}$ is a security parameter, it generates a master public/secret key pair (mpk, msk).
2. PartialKeyGen (User Partial Key Generation): On input msk and user identity $ID \in \{0,1\}^*$, it generates a key $partial_key$ called user partial key.

3. UserKeyGen (User Key Generation): On input mpk and user identity ID, it generates a user public/secret key pair (upk, usk).
4. CL-Sign (Certificateless Signature Generation): On input user secret key usk, user partial key $partial_key$ and message $m \in \{0,1\}^*$, it generates a signature σ.
5. CL-Ver (Certificateless Signature Verification): On input mpk, user identity ID, user public key upk, message m and signature σ, it returns accept or reject.

For correctness, we require that for all $k \in \mathbb{N}$, $m \in \{0,1\}^*$, $ID \in \{0,1\}^*$, if $(mpk, msk) \leftarrow \mathsf{MasterKeyGen}(1^k)$, $partial_key \leftarrow \mathsf{PartialKeyGen}(msk, ID)$, $(upk, usk) \leftarrow \mathsf{UserKeyGen}(mpk, ID)$, and $\sigma \leftarrow \mathsf{CL\text{-}Sign}(usk, partial_key, m)$, then $\mathsf{CL\text{-}Ver}(mpk, ID, upk, m, \sigma) = 1$.

Case of Invalid Input: For each of the algorithms above, we implicitly assume that there is a domain for each of its inputs. An input is said to be *valid* if it falls in its corresponding domain. For example, the domain of msk is defined by the set of all possible output values of master secret key of MasterKeyGen for each given security parameter $k \in \mathbb{N}$. Hence if any of the inputs of an algorithm above is invalid, then the algorithm will output a symbol $\perp$ indicating that the execution of the algorithm is halted with failure.

In practice, the KGC (Key Generation Center) could be the one who performs MasterKeyGen and PartialKeyGen. The master public key mpk will then be published and assumed that everyone in the system has got a legitimate copy of it. The partial key is also assumed to be issued securely to the intended user so that no one except the intended user can get the partial key. For each user in the system, the user is supposed to be able to carry out UserKeyGen, CL-Sign and CL-Ver. It is the user's responsibility to forward the user public key upk to the intended receiver(s) and announces the identity ID.

In the rest of the paper, we sometimes denote the user partial key of a user with identity ID as $partial_key_{ID}$ and the user public/secret key pair as (upk_{ID}, usk_{ID}). The subscript of ID in the notations is simply for helping the presentation of the paper.

2.1 Differences from the Old Definition

There are several differences between the new definition above and the previously adopted definition, which was first introduced in [1], later adopted by [11, 8]. In the old definition, there are seven algorithms defined. However, we notice that in the old definition, besides the signature verification algorithm, there are two more deterministic algorithms defined solely for generating the user public key upk and combining the partial key $partial_key$ with the user secret key usk. In the following, we show that the functions of these two algorithms can simply be included in other algorithms without loss of functionality.

The old definition has a user **private** key generated deterministically from $partial_key$ and usk. This user private key is then used solely in signature generation. By including the function of user private key generation into the

signature generation algorithm, we eliminate the user private key generation algorithm. The signature generation algorithm is now taking both *partial_key* and *usk* as inputs. To see that this new approach captures the original definition, we can consider the new signature generation algorithm to have two phases. In the first phase, the original user private key generation algorithm is executed and a user private key is generated. In the second phase, the original signature generation algorithm is executed by taking the user private key as one of its inputs. This new definition is more versatile as it is possible that *partial_key* and *usk* are 'mixed' together in some randomized way during signature generation.

Another difference is that in the original definition, there is an additional algorithm to deterministically generate the user public key *upk* from *mpk*, user identity ID and user secret key *usk*. In the definition above, we combine the function of this algorithm with UserKeyGen. We can see that this new definition is more versatile as it also includes schemes which may have multiple possible values for *upk* with respect to the same value of *usk*. It also retains the important feature of certificateless cryptography, that is, user partial key generation and user key generation can be done *independently* by the KGC and the user, respectively. In particular, a user with identity ID can generate upk_{ID} even before the KGC generates $partial_key_{ID}$ for the user.

In [11], PartialKeyGen is restricted to be deterministic. This implies that for each user identity ID, there is exactly one partial key corresponding to ID. Although, to our knowledge, all previously proposed certificateless signature schemes [1, 8] use deterministic algorithms for PartialKeyGen, it is too restrictive to require that a generic construction of certificateless signature should also use a deterministic PartialKeyGen. When comparing with its counterpart, the identity-based signature, there are schemes [2] where the key generation server uses randomized algorithm to generate keys for its users, we propose to relax the restriction by allowing the algorithm to be randomized, that is, different values of partial key may be obtained by running PartialKeyGen for multiple times, each time with fresh random coins.

2.2 Adversarial Model

There are two types of adversaries, $\mathcal{A}_I$ and $\mathcal{A}_{II}$. Adversary $\mathcal{A}_I$ simulates attacks when the adversary (or signature forger) compromises the user secret key *usk* or replaces the user public key *upk*. However, $\mathcal{A}_I$ is not given the master secret key *msk* nor getting access to the user partial key *partial_key*. Adversary $\mathcal{A}_{II}$ simulates attacks when the adversary knows *msk* and *partial_key*. But $\mathcal{A}_{II}$ can no longer get access to *usk* nor replace *upk*.

Informally, $\mathcal{A}_I$ represents a third party who may compromise the target user's *usk* or replace *upk*, but does not get access to the user's *partial_key* nor *msk* of the KGC. $\mathcal{A}_{II}$ models an *eavesdropping* KGC or a *colluder* of the KGC, who knows *msk* and can derive the value of any user's *partial_key*, but cannot obtain *usk* or replace *upk*. As remarked in [1], $\mathcal{A}_{II}$ models an *eavesdropping* KGC who does not replace user public keys. We adopt their notion in our paper

but also emphasize that the KGC should also be assumed to have generated msk according to the scheme specification. Below are the details.

There are five oracles which can be accessed by the adversaries according to the game specifications which will be given shortly. The five oracles are:

1. CreateUser: On input an identity $ID \in \{0,1\}^*$, if ID has already been created, nothing is to be carried out by the oracle. Otherwise, the oracle generates $partial_key_{ID} \leftarrow \mathsf{PartialKeyGen}(msk, ID)$ and $(upk_{ID}, usk_{ID}) \leftarrow \mathsf{UserKeyGen}(mpk, ID)$. In this case, ID is said to be *created*. In both cases, upk_{ID} is returned.
2. RevealPartialKey: On input an identity ID, it returns $partial_key_{ID}$ if ID has been created. Otherwise, a symbol $\perp$ is returned.
3. RevealSecretKey: On input an identity ID, it returns the corresponding user secret key usk_{ID} if ID has been created. Otherwise, a symbol $\perp$ is returned.
4. ReplaceKey: On input an identity ID and a user public/secret key pair (upk^*, usk^*), the original user public/secret key pair of ID is replaced with (upk^*, usk^*) if ID has been created. Otherwise, no action will be taken.
5. Sign: On input an identity ID and a message $m \in \{0,1\}^*$, the signing oracle returns a valid signature σ if ID has been created but the user public/secret key pair (upk_{ID}, usk_{ID}) has not been replaced. The signature σ is *valid* if $\mathsf{CL\text{-}Ver}(mpk, ID, upk_{ID}, m, \sigma) = 1$. If ID has not been created, a symbol $\perp$ is returned. If the user public/secret key pair of ID has been replaced with, say (upk^*, usk^*), then the oracle returns the result of $\mathsf{CL\text{-}Sign}(usk^*, partial_key_{ID}, m)$.

Remark: When querying the oracle ReplaceKey, usk^* can be an empty string. In this case, it means that the user secret key is not provided. If usk^* is an empty string and the original user secret key of an identity ID is replaced with usk^*, then the empty string will be returned if the RevealSecretKey oracle is queried on ID. Also note that even if usk^* is not an empty string, it does not mean that usk^* is the corresponding secret key of upk^*. Hence as mentioned, the signature generated by the signing oracle Sign will be an execution of CL-Sign using the replaced user secret key usk^* regardless of the value of upk^*. In other words, the signature may not be valid. Also check page 238, "*Case of Invalid Input*", for the handling of invalid usk^*.

We define two games, one for $\mathcal{A}_I$ and the other one for $\mathcal{A}_{II}$.

Game I: Let $\mathcal{S}_I$ be the game simulator/challenger and $k \in \mathbb{N}$ be a security parameter.

1. $\mathcal{S}_I$ executes $\mathsf{MasterKeyGen}(1^k)$ to get (mpk, msk).
2. $\mathcal{S}_I$ runs $\mathcal{A}_I$ on 1^k and mpk. During the simulation, $\mathcal{A}$ can make queries onto CreateUser, RevealPartialKey, RevealSecretKey, ReplaceKey and Sign.
3. $\mathcal{A}_I$ is to output (ID^*, m^*, σ^*).

$\mathcal{A}_I$ *wins* if $\mathsf{CL\text{-}Ver}(mpk, ID^*, upk_{ID^*}, m^*, \sigma^*) = 1$ for some created ID^* and the oracle Sign has never been queried with (ID^*, m^*). One **additional restriction** is that $\mathcal{A}_I$ has never queried $\mathsf{RevealPartialKey}(ID^*)$ to get the user partial key $partial_key_{ID^*}$.

A certificateless signature scheme is secure in **Game I** if for all probabilistic polynomial-time (PPT) algorithm $\mathcal{A}_I$, it is negligible for $\mathcal{A}_I$ to win the game. Note that $\mathcal{A}_I$ may have queried $\mathsf{RevealSecretKey}(ID^*)$ and got the user secret key usk_{ID^*} or queried $\mathsf{ReplaceKey}(ID^*, \cdot, \cdot)$ and replaced the user public key upk_{ID^*} before generating a forgery (ID^*, m^*, σ^*).

Game II: In this game, the simulator $\mathcal{S}_{II}$, similar to $\mathcal{S}_I$ in **Game I** above, interacts with the other adversary $\mathcal{A}_{II}$ in almost the same way, except the following differences.

- When start running $\mathcal{A}_{II}$, besides giving mpk to $\mathcal{A}_{II}$, $\mathcal{S}_{II}$ also gives msk to $\mathcal{A}_{II}$.
- The additional restriction is changed by requiring that $\mathcal{A}_{II}$ has never queried $\mathsf{RevealSecretKey}(ID^*)$ to get the user secret key usk_{ID^*} nor queried $\mathsf{ReplaceKey}(ID^*, \cdot, \cdot)$ to replace the user public key upk_{ID^*}.

A certificateless signature scheme is secure in **Game II** if for all PPT algorithm $\mathcal{A}_{II}$, it is negligible for $\mathcal{A}_{II}$ to win the game.

2.3 Discussions

Comparing with previously proposed adversarial models [1, 11, 8][1], one significant difference is that all previous models do not allow $\mathcal{A}_{II}$ to query $\mathsf{ReplaceKey}$ (in **Game II**) at any point. Of course, we have to assume that $\mathcal{A}_{II}$ does not mount an attack against a user with identity ID when the user public key upk_{ID} is also replaced by $\mathcal{A}_{II}$ because $\mathcal{A}_{II}$ can always get the user partial key $partial_key_{ID}$. However, allowing $\mathcal{A}_{II}$ to query $\mathsf{ReplaceKey}$ with identities other than ID does not boost $\mathcal{A}_{II}$'s attacking capability either. This is because $\mathcal{A}_{II}$ can always obtain all the secrets corresponding to identities other than ID. Hence to ease peer-to-peer comparison between the attacking powers of $\mathcal{A}_I$ and $\mathcal{A}_{II}$, we choose to provide the same set of oracles to both of the adversaries.

Another difference is that in all the previous models, the additional restriction mentioned in **Game I** above is that *either*

- $\mathcal{A}_I$ has never queried $\mathsf{RevealPartialKey}(ID^*)$ to get the user partial key $partial_key_{ID^*}$; *or*
- $\mathcal{A}_I$ has never queried $\mathsf{RevealSecretKey}(ID^*)$ to get the user secret key usk_{ID^*} nor queried $\mathsf{ReplaceKey}(ID^*, \cdot, \cdot)$ to replace the user public key upk_{ID^*}.

This modeling approach provides a direct association between $\mathcal{A}_I$ and the third party (also known as an outsider) in the "real world", because the third party may manage to either replace the user public key or compromise the user partial key. In our model, instead of focusing on providing the direct association between $\mathcal{A}_I$ and the type of attackers in the real world, we focus on simplifying the model so that it may lead to simpler proofs of security. In particular, in our model

[1] Al-Riyami and Paterson did not really develop a full security model for certificateless signature in [1]. It was later in [11, 8] that more formalized and complete models were specified.

instead, the additional restriction above is *simplified* to just requiring that $\mathcal{A}_I$ has never queried RevealPartialKey(ID^*). We do not consider the case that $\mathcal{A}_I$ reveals $partial_key_{ID^*}$ but focus on the case that $\mathcal{A}_I$ compromises usk_{ID^*} or replaces upk_{ID^*}.

This simplification does not weaken our model because attacks concerning revealed $partial_key_{ID^*}$ have already been captured in **Game II** of our model. To see this, consider that $\mathcal{A}_{II}$ does not use msk in the entire simulation but has launched RevealPartialKey(ID^*) at some point before generating a forgery. In this way, $\mathcal{A}_{II}$ can simulate all the attacks captured in **Game I** of all previous models for the case that RevealPartialKey(ID^*) is queried.

To move one more step forward, we can see that our simplified model is even stronger than all the previous models. In **Game I** of all the previous models, $\mathcal{A}_I$ is not allowed to just compromise usk_{ID^*} without compromising $partial_key_{ID^*}$. In fact, there is no dedicated oracle in any of the previous models that allows $\mathcal{A}_I$ to compromise just the user secret key. Instead, they only have one oracle which returns both the user partial key and the user secret key at the same time once queried. Obviously, the oracle should not be queried on ID^* by $\mathcal{A}_I$. Hence, it is impossible for previous models to capture attacks when usk_{ID^*} is compromised while $partial_key_{ID^*}$ is not. In our simplified model above, we allow $\mathcal{A}_I$ to compromise a user secret key via RevealSecretKey. Note that a certificateless signature scheme should still be secure even if usk_{ID^*} is compromised provided that $partial_key_{ID^*}$ is not. This is one of the major attacking scenarios that our **Game I** can capture while all the previous models cannot.

3 Cryptanalysis of Yum-Lee Generic Construction

In [11], Yum and Lee proposed a generic construction of certificateless signature. The generic construction is based on two building blocks: a conventional signature scheme and an identity-based signature (IBS) scheme [10, 2]. The conventional signature scheme is assumed to be existential unforgeable against chosen message attack (euf-cma) in the sense of [7] and the IBS scheme is assumed to be existential unforgeable against chosen message attack and ID attacks (euf-cma-ida) [2]. Below is the review of Yum-Lee generic certificateless signature scheme.

Let $\Pi_{PK} = (Gen_{PK}, Sign_{PK}, Ver_{PK})$ be an euf-cma secure signature scheme, where Gen_{PK} takes a security parameter and generates a public/secret key pair denoted by (pk_{PK}, sk_{PK}); $Sign_{PK}$ takes a private signing key and a message and generates a signature denoted by σ_{PK}; and Ver_{PK} is the corresponding signature verification algorithm.

Let $\Pi_{IBS} = (Gen_{IBS}, UKGen_{IBS}, Sign_{IBS}, Ver_{IBS})$ be an euf-cma-ida secure IBS scheme, where Gen_{IBS} takes a security parameter and generates an identity-based master public/secret key pair denoted by (mpk_{IBS}, msk_{IBS}); $UKGen_{IBS}$ is the user-key generation algorithm which takes msk_{IBS} and an identity ID and generates a secret key denoted by $sk_{IBS}[ID]$; $Sign_{IBS}$ takes $sk_{IBS}[ID]$ and a message and generates a signature denoted by σ_{IBS}; and Ver_{IBS} is the corresponding signature verification algorithm.

As mentioned in Sec. 2.1, there are seven algorithms for a certificateless signature scheme. This is also the case for Yum-Lee generic construction. By following the discussions in Sec. 2.1, we can easily convert them into the following five algorithms which conform to the simplified definition given in Sec. 2.

$(mpk, msk) \leftarrow$ **MasterKeyGen**(1^k)
Run $(mpk_{IBS}, msk_{IBS}) \leftarrow Gen_{IBS}(1^k)$; and
set $mpk := mpk_{IBS}$ and $msk := msk_{IBS}$.

$partial_key_{ID} \leftarrow$ **PartialKeyGen**(msk, ID)
Run $sk_{IBS}[ID] \leftarrow UKGen_{IBS}(msk, ID)$; and
set $partial_key_{ID} := \langle\ sk_{IBS}[ID] \parallel ID\ \rangle$.

$(upk_{ID}, usk_{ID}) \leftarrow$ **UserKeyGen**(mpk, ID)
Run $(pk_{PK}, sk_{PK}) \leftarrow Gen_{PK}(1^k)$; and
set $upk_{ID} := pk_{PK}$ and $usk_{ID} := sk_{PK}$.

$\sigma \leftarrow$ **CL-Sign**$(usk_{ID}, partial_key_{ID}, m)$
Run $\sigma_{PK} \leftarrow Sign_{PK}(usk_{ID}, m)$;
set $m' := \langle\ \sigma_{PK} \parallel ID\ \rangle$;
run $\sigma_{IBS} \leftarrow Sign_{IBS}(sk_{IBS}[ID], m')$; and
set $\sigma := \langle\ \sigma_{PK} \parallel \sigma_{IBS}\ \rangle$.

$1/0 \leftarrow$ **CL-Ver**$(mpk, ID, upk_{ID}, m, \sigma)$
Parse σ into $\langle\ \sigma_{PK} \parallel \sigma_{IBS}\ \rangle$;
run $b_1 \leftarrow Ver_{IBS}(mpk, ID, \langle \sigma_{PK} \| ID \rangle, \sigma_{IBS})$;
run $b_2 \leftarrow Ver_{PK}(upk_{ID}, m, \sigma_{PK})$; and
set output to $b_1 \wedge b_2$.

In the description above, $\langle X \rangle$ represents some encoding of X for binary representation; the symbol $\|$ represents binary string concatenation; and $\wedge$ represents bitwise AND operation.

3.1 Key Replacement Attack

In [11, Sec. 3.2, Theorem 1], the authors claimed that the scheme above is secure in **Game I** and **Game II** against $\mathcal{A}_I$ and $\mathcal{A}_{II}$, respectively, under the conditions that Π_{PK} is euf-cma secure and Π_{IBS} is euf-cma-ida secure.

However, we find that these two conditions are insufficient to secure their certificateless signature scheme reviewed above. In particular, we will show below that their scheme is insecure in **Game I** against $\mathcal{A}_I$. By replacing the user public key of the target user, $\mathcal{A}_I$ is able to forge successfully. Below are the details.

Idea of Attack: For a conventional signature scheme Π_{PK}, if it is euf-cma, it means that an adversary who knows only the public key pk_{PK} but not the corresponding secret key sk_{PK} is not able to forge a message-signature pair (m, σ) such that $Ver_{PK}(pk_{PK}, m, \sigma) = 1$, even if the adversary is allowed to adaptively query a signing oracle with respect to (pk_{PK}, sk_{PK}). However, euf-cma security does not guarantee that given a valid message-signature pair (m, σ)

with respect to (pk_{PK}, sk_{PK}), the adversary is not able to come up with a triple (σ, m', pk') such that $m' \neq m$, $pk' \neq pk_{PK}$ but $Ver_{PK}(pk', m', \sigma) = 1$. On the contrary, this '**Message-And-Key-Replacement Attack**' is not unusual in proven euf-cma secure signature schemes. Two concrete examples are to be shown in the next section.

Attack Details: We now show that in **Game I**, the adversary $\mathcal{A}_I$ can win the game by making use of the message-and-key-replacement attack.

1. $\mathcal{A}_I$ arbitrarily picks an identity $ID \in \{0,1\}^*$ and queries $\mathsf{CreateUser}(ID)$ to 'create' the corresponding user. Suppose the returned user public key is pk_{PK};
2. $\mathcal{A}_I$ then arbitrarily picks a message $m \in \{0,1\}^*$ and queries $\mathsf{Sign}(ID, m)$. Suppose the signature returned by the oracle is $\sigma = \langle\ \sigma_{PK} \parallel \sigma_{IBS}\ \rangle$;
3. $\mathcal{A}_I$ generates a triple (σ, m', pk') such that $m' \neq m$, $pk' \neq pk_{PK}$, and $Ver_{PK}(pk', m', \sigma_{PK}) = 1$.
4. $\mathcal{A}_I$ queries $\mathsf{ReplaceKey}(ID, pk', \lambda)$ to change the public key of ID, where λ denotes an empty string which corresponds to the replacing user secret key.
5. Finally, $\mathcal{A}_I$ outputs (ID, m', σ).

Note that the signing oracle Sign has never been queried with (ID, m') and obviously σ is a valid signature for message m' of ID as the user public key is now equal to pk'. The additional restriction stated in **Game I** is also satisfied.

3.2 Examples of Message-And-Key-Replacement Attack Against EUF-CMA Secure Signature Schemes

The first example is the hash-based ElGamal signature scheme [6, 9]. The hash-based version [9] has been shown to be euf-cma under the random oracle model [3]. The second example is a pairing-based signature scheme due to Zhang et al. [12] which has been shown to be euf-cma also under the random oracle model. However, due to page limitation, the description of the second example has to be omitted. Please refer to the full version of this paper for details.

Example 1: Hash-Based ElGamal Signature Scheme. Let p and q be two large prime such that $q|p-1$. Let g be an element of $\mathbb{Z}_p^*$ of order q. Suppose (pk, sk) be a public/secret key pair where $pk = (p, q, g, y)$, $sk \in_R \mathbb{Z}_q$ and $y = g^{sk} \bmod p$. To sign a message $m \in \{0,1\}^*$, the following steps are carried out.

1. Randomly pick $\ell \in_R \mathbb{Z}_q$ and compute $r = g^{\ell} \bmod p$;
2. Set $e \leftarrow H(m, r)$ where $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a hash function;
3. Compute $s = \ell^{-1}(e - sk \cdot r) \bmod q$; and
4. Set the signature $\sigma = (r, s)$.

To verify, check if $r \in \mathbb{Z}_p^*$ and whether $y^r r^s \stackrel{?}{\equiv} g^{H(m,r)} \pmod p$. If all of them are correct, output 1 for accepting the signature; otherwise, output 0.

Message-And-Key-Replacement Attack: Given message m, signature $\sigma = (r, s)$ and public key $pk = (p, q, g, y)$, the attack can be launched as follows.

1. Randomly pick a message $m' \in \{0,1\}^*$ such that $m' \neq m$;
2. Compute $y' = yg^{(H(m',r)-H(m,r))/r} \bmod p$ and set a new public key $pk' = (p, q, g, y')$; and
3. Output a triple (σ, m', pk').

The attack is said to be successful if the signature verification algorithm returns 1. We can see that the verification will pass because

$$\begin{aligned} y'^r r^s &\equiv (yg^{(H(m',r)-H(m,r))/r})^r r^s \\ &\equiv y^r g^{H(m',r)-H(m,r)} g^{H(m,r)-sk\cdot r} \\ &\equiv g^{H(m',r)} \pmod{p}. \end{aligned}$$

Discussions. We should emphasize that the possession of message-and-key-replacement property does not constitute a weakness of a signature scheme. The security goal of a signature scheme we are dealing with is still the existential unforgeability against chosen message attack (euf-cma) [7]. Message-and-key-replacement attack is also reminiscent of the *duplicate-signature key selection attack* [4]: Given a valid message-signature pair (m, σ) under a public key pk, the attack is to find *another* public/secret key pair (pk', sk') such that (m, σ) is also a valid pair under pk'. Note that in the duplicate-signature key selection attack, it is required to derive a new secret key sk', while in the message-and-key-replacement attack, this is not a requirement. Also, it is not necessary for a duplicate-signature key selection attacker to alter the message m, while in its counterpart, this is a requirement. An interesting observation is that some of the techniques for launching duplicate-signature key selection attack described in [4] can actually be extended for launching the message-and-key-replacement attack, for example, against a hash-based RSA signature scheme.

4 An Improved Generic Construction

Our modification changes the signature generation algorithm as follows.

$\sigma \leftarrow$ **CL-Sign**$(usk_{ID}, partial_key_{ID}, m)$
Set $m' := \langle\ m \parallel mpk \parallel ID \parallel upk_{ID}\ \rangle$;
run $\sigma_{PK} \leftarrow Sign_{PK}(usk_{ID}, m')$;
set $m'' := \langle\ m \parallel mpk \parallel ID \parallel upk_{ID} \parallel \sigma_{PK}\ \rangle$;
run $\sigma_{IBS} \leftarrow Sign_{IBS}(sk_{IBS}[ID], m'')$; and
set $\sigma := \langle\ \sigma_{PK} \parallel \sigma_{IBS}\ \rangle$.

The signature verification algorithm is then changed accordingly by checking that both the identity-based signature verification algorithm Ver_{IBS} and the conventional signature verification algorithm Ver_{PK} will return 1 on the corresponding messages m'' and m', respectively.

Theorem 1. *The modified generic construction is secure in* ***Game I*** *(as defined in Sec. 2.2) if the IBS scheme* $\Pi_{IBS} = (Gen_{IBS}, UKGen_{IBS}, Sign_{IBS}, Ver_{IBS})$ *is* euf-cma-ida *secure [2].*

Due to page limitation, the proof has to be omitted. Please refer to the full version of this paper for details.

Theorem 2. *The modified generic construction is secure in* ***Game II*** *if the signature scheme* $\Pi_{PK} = (Gen_{PK}, UKGen_{PK}, Sign_{PK}, Ver_{PK})$ *is* euf-cma *secure [7].*

Due to page limitation, the proof has to be omitted. Please refer to the full version of this paper for details.

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

1. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *Proc. ASIACRYPT 2003*, pages 452–473. Springer-Verlag, 2003. LNCS 2894.
2. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In *Proc. EUROCRYPT 2004*, pages 268–286. Springer-Verlag, 2004. LNCS 3027 (Full paper is available at Bellare's homepage URL: `http://www-cse.ucsd.edu/users/mihir`).
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM.
4. S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99*, pages 154–170. Springer-Verlag, 1999. LNCS 1560.
5. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Proc. CRYPTO 2001*, pages 213–229. Springer-Verlag, 2001. LNCS 2139.
6. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31(4):469–472, 1985.
7. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, April 1988.
8. X. Huang, W. Susilo, Y. Mu, and F. Zhang. On the security of certificateless signature schemes from Asiacrypt 2003. In *Cryptology and Network Security, 4th International Conference, CANS 2005*, pages 13–25. Springer-Verlag, 2005. LNCS 3810.
9. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proc. EUROCRYPT 96*, pages 387–398, 1996. LNCS 1070.
10. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. CRYPTO 84*, pages 47–53. Springer, 1984. LNCS 196.
11. D. H. Yum and P. J. Lee. Generic construction of certificateless signature. In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004*, pages 200–211. Springer-Verlag, 2004. LNCS 3108.
12. F. Zhang, R. Safavi-Naini, and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In *7th International Workshop on Theory and Practice in Public Key Cryptography (PKC 2004)*, pages 277–290. Springer, 2004. LNCS 2947.

A Novel Range Test

Kun Peng[1], Colin Boyd[1], Ed Dawson[1], and Eiji Okamoto[2]

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

Abstract. In a range test, one party holds a ciphertext and needs to test whether the message encrypted in the ciphertext is within a certain interval range. In this paper, a range test protocol is proposed, where the party holding the ciphertext asks another party holding the private key of the encryption algorithm to help him. These two parties run the protocol to implement the test. The test returns TRUE if and only if the encrypted message is within the certain interval range. If the two parties do not conspire, no information about the encrypted message is revealed from the test except what can be deduced from the test result. Advantages of the new protocol over the existing related techniques are that it achieves correctness, soundness, flexibility, high efficiency and privacy simultaneously.

Keywords: interval range, range test, specialized zero test, correctness, soundness.

1 Introduction

In a range test, one party (the *tester*) holds a ciphertext and needs to test whether the message encrypted in the ciphertext is within a certain interval range. This test is frequently required in cryptographic applications like e-auction [8], electronic voting [11], electronic finance [4], group signature [3], publicly verifiable secret sharing [7] and verifiable encryption [1]. Four properties are desired in a range test. Correctness: If the encrypted message is in the interval range, the test outputs TRUE. Soundness: If the test outputs TRUE, the encrypted message is in the interval range. Privacy: No information about the encrypted message is revealed except what can be deduced from the test result. Flexibility: The limitation on the range size, encryption format and participants should be as little as possible.

The simplest way to implement a range test is using multiple equality tests linked by "OR" logic without revealing which number in the range equals the encrypted message. Currently, there are two methods (called naive range test in this paper) to implement this idea: zero knowledge proof of "OR" logic by Cramer *et al* [5] or the verification technique called zero test [10]. These two methods can be flexibly employed so that various ranges (e.g. ranges with very

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 247–258, 2006.
© Springer-Verlag Berlin Heidelberg 2006

large size), participant models (with or without prover) and encryption formats (even commitment formats) can be used. Although naive range test can be flexible, correct, sound and private, it is very inefficient as its cost is linear in the size of the range. If the ciphertext to test is encrypted in some special encryption format (e.g. encrypted bit by bit), cost of naive range test can be reduced to be linear in the logarithm of the range size. However, ciphertext in practical cryptographic applications (especially when secure computation of ciphertext is needed) cannot be often encrypted in special encryption format. So for the sake of flexibility, naive range test generally needs a cost linear in the size of the range. Even if the special encryption format can be employed to improve efficiency, naive range test is still too inefficient.

Some cryptographic techniques [1, 2, 7, 4] are related to range test. They prove that a committed message is within a certain interval range and are called RPC (range proof of commitment) schemes in this paper. In RPC schemes, a prover with the knowledge of the committed message is needed to give a zero knowledge proof that the message is in the certain interval range. Although RPC schemes are efficient as their cost is independent of the size of the range, they have some drawbacks. Firstly, in many applications like e-auction and e-voting, encrypted messages instead of committed messages are required to be tested. So RPC schemes (especially [2], which requires a certain commitment format) cannot be employed in these applications. Secondly, the message to be tested may be generated by multiple parties and unknown to anybody. For example, in the k^{th}-bid auction [8], the seller has to test whether the number of bids at a price is less than k without revealing the bids. As no single bidder knows the sum, nobody can provide any proof to implement the test. In another example, e-banking, it is required to test whether a sum of money is below a threshold without revealing it while nobody knows the sum as it accumulates multiple dealings. So a prover is not always available. Thirdly, most RPC schemes [1, 7, 4] cannot guarantee correctness and soundness at the same time. The only correct and sound scheme among them is Boudot [2], which is only asymptotically (instead of absolutely) sound. Finally, all the known RPC schemes can work only when the range to test is many magnitudes smaller than the size of the message space of the commitment algorithm. In theory, solution to the millionaire problem and solution to range test can be reduced to each other. However, the existing solution to the millionaire problem [12, 10] are either inefficient [10] or not completely private [12].

As the drawbacks of RPC schemes and millionaire problem schemes limit their application, a range test protocol is proposed in this paper, which is much more efficient than naive range test and overcomes the drawbacks of the RPC schemes. In the new range test protocol, two parties are involved: a tester and a (decryption) authority, who can be acted by multiple entities through a threshold key sharing mechanism. The tester holds the ciphertext to test. The private key to decrypt the ciphertext is held by the authority. So the tester asks the authority for help and they run the protocol to implement the test. If the encrypted message is in the certain interval range, the protocol outputs TRUE.

If the encrypted message is not in the certain interval range, the protocol outputs FALSE. Namely, the new test protocol is correct and sound. If the two parties do not conspire, no information about the encrypted message is revealed from the test except what can be deduced from the test result. The new protocol is flexible as it accepts ranges of the same magnitude as the size of the message space of the encryption algorithm and does not need any prover with knowledge of the encrypted message. The new protocol is efficient as its computational cost is independent of the range size. This new protocol can overcome the drawbacks of RPC schemes. In the example of k^{th}-bid auction, the seller acts as the tester while an auctioneer acts as the authority to help the seller to determine whether the number of bids at a price is over k without revealing the bids or the number. In the example of e-banking, two servers (neither knowing the sum of the money) act as the tester and authority to test the range of the sum. If the two servers do not conspire, the sum is not revealed.

In this paper secuirty is analysed in the negatively-malicious model: the adversary does not deviate from the protocol in his attack. The structure of this paper is as follows. Parameters and symbols to be used in the paper are defined in Section 2. In Section 3, a building block, specialized zero test, is designed. In Section 4, two range test protocols are proposed. They are not independent. Instead, the second protocol is an optimization of the first one.

2 Preliminary Work

Parameters, symbols and encryption systems to be used later are described in this section. Two additive homomorphic semantically-secure encryption systems[1] (e.g. modified ElGamal encryption [6]) are needed in this paper. They are called the first encryption system and the second encryption system respectively later in this paper. The ciphertext to test is encrypted in the first encryption system, while the tester holds the ciphertext and the authority holds the private key of the first encryption system. To implement the range test, a second encryption system is set up and its private key is also held by the authority. The public keys of both encryption systems are public, so that both the authority and the tester can use both encryption systems for encryption. The message spaces of the two encryption systems are Z_{p_1} and Z_{p_2} respectively. It is required in this paper that $p_2 \geq 3p_1$ and p_2 is a prime.

Although any additive homomorphic semantically-secure encryption algorithm like Paillier encryption [9] can be employed, for simplicity the modified ElGamal encryption [6] is employed in both encryption systems in this paper.

[1] An encryption algorithm with message space Z_p and decryption function $D()$ is additive homomorphic if $D(c_1) + D(c_2) = D(c_1 c_2) \bmod p$ for any ciphertexts c_1 and c_2. An encryption algorithm is semantically-secure if given a ciphertext c, two messages m_1, m_2 and querry to the encryption function, such that $c = E(m_i)$ where $i = 1$ or 2, there is no polynomial algorithm to find out i.

In this manner, p_1 and p_2 are both large prime numbers. Details about the two (modified ElGamal) encryption systems are as follows.

- Multiplication in the two encryption systems are computed with modulus p'_1 and p'_2 respectively.
- $< g_1 >$ and $< g_2 >$ are cyclic subgroups of $Z^*_{p'_1}$ and $Z^*_{p'_2}$ respectively with generator g_1 and g_2, which have prime order p_1 and p_2 respectively.
- The message space in the two encryption systems are Z_{p_1} and Z_{p_2} respectively.
- $x_1 \in Z_{p_1}$ and $x_2 \in Z_{p_2}$ are private keys of the two encryption systems respectively. (g_1, y_1) and (g_2, y_2) are public keys of the two encryption systems respectively where $y_1 = g_1^{x_1} \bmod p'_1$ and $y_2 = g_2^{x_2} \bmod p'_2$.
- $E_1(m)$ and $E_2(m)$ stand for encryption of message m in the two encryption systems respectively: $(g_1^r \bmod p'_1, g_1^m y_1^r \bmod p'_1)$ or $(g_2^r \bmod p'_2, g_2^m y_2^r \bmod p'_2)$ where r is randomly chosen from Z_{p_1} or Z_{p_2}.
- The product of two ciphertexts $c_1 = (a_1, b_1)$ and $c_2 = (a_2, b_2)$ in the two encryption systems is $(a_1 a_2, b_1 b_2)$. Inversion of a ciphertext $c = (a, b)$ is (a^{-1}, b^{-1}). With multiplication and inversion defined, definition of exponentiation and division is automatically obtained.
- $D_1(c)$ and $D_2(c)$, decryption function of ciphertext $c = (a, b)$ in the two encryption systems respectively, is $\log_{g_1} b/a^{x_1}$ and $\log_{g_1} b/a^{x_1}$ respectively. Although normally decryption in the modified ElGamal encryption algorithm needs a search of logarithm and is not efficient, it is only required to test whether the message is zero or not in any decryption in this paper, which does not need any logarithm search and is very efficient.

Later in this paper, encryption, decryption, ciphertext multiplication, ciphertext inversion and ciphertext exponentiations are computed as described here in this section. The other symbols to be used in this paper are listed in Table 1.

Table 1. Symbols

$a\%b$	outputs an integer c smaller than b such that $a = c \bmod b$
$\|a\|$	the absolute value of an integer a
$[S]$	the size of a set S
$\binom{a}{b}$	the number of possible choices of b elements from a candidate elements

3 A Building Block — Specialized Zero Test

Zero test is a technique to test whether there is at least one null ciphertext (encryption of zero) among multiple ciphertexts. A zero test must be private, namely nothing about the messages encrypted in the ciphertexts can be deduced from the test except whether there is at least one null ciphertext among them. The existing zero test technique (e.g. the so-called complex zero test in [10]) cannot obtain complete privacy as it may reveal some information about the number of null ciphertexts. Fortunately, in this paper it is only desired to test

whether there is one null ciphertext among multiple ciphertexts where there is at most one null ciphertext among them. This will be accomplished by modifying the zero test technique from [10] into a new cryptographic primitive: *specialized zero test*, which can achieve complete privacy in the application in this paper. A specialized zero test examines whether there is one null ciphertext among multiple ciphertexts encrypted using the second encryption system described in Section 2 where there is at most one null ciphertext among them. While the zero test technique in [10] is a multiparty protocol, only two parties are involved in the specialized zero test in this paper: a tester A_1 and an authority A_2. A_1 holds ciphertexts $c_1, c_2, \ldots, c_n$ in the second encryption system where there is at most one null ciphertext among them. A_2 holds the private key of the second encryption system. In the specialized zero test A_2 assists A_1 to test whether there is one null ciphertext among $c_1, c_2, \ldots, c_n$. Three properties are desired in specialized zero test.

- Correctness: if there is one null ciphertext in $c_1, c_2, \ldots, c_n$, the test result is TRUE.
- Soundness: if the test result is TRUE, there is one null ciphertext in $c_1, c_2, \ldots, c_n$.
- Privacy: after the test, each party learns only the test result and what can be deduced from it, as long as the authority and the tester do not collude.

The test protocol is denoted as $ZM(A_1,\ A_2 \mid c_1, c_2, \ldots, c_n)$ and described in Figure 1.

1. A_1 chooses $\pi()$, a permutation on $\{1, 2, \ldots, n\}$, and random integers r_i from $Z_{p_2} - \{0\}$ for $i = 1, 2, \ldots, n$. Then he calculates $c'_i = c^{r_i}_{\pi(i)}$ for $i = 1, 2, \ldots, n$. He sends $c'_1, c'_2, \ldots, c'_n$ to A_2.
2. A_2 calculates $d_i = D_2(c'_i)$ for $i = 1, 2, \ldots, n$ one by one until one d_i is found to be zero or all the n ciphertexts are decrypted. A_2 publishes the output of the zero test as follows.

$$ZM(A_1,\ A_2 \mid c_1, c_2, \ldots, c_n) = \begin{cases} \mathsf{TRUE} & \text{if zero found in } d_i \\ \mathsf{FALSE} & \text{if no zero in } d_i \end{cases} \quad (1)$$

Fig. 1. Specialized zero test

Theorem 1. *The specialized zero test is correct in the negatively-malicious model. More precisely, if nobody deviates from the protocol and there is one zero encrypted in $c_1, c_2, \ldots, c_n$, then $ZM(A_1,\ A_2 \mid c_1, c_2, \ldots, c_n) = \mathsf{TRUE}$.*

Proof: As $c'_i = c^{r_i}_{\pi(i)}$ for $i = 1, 2, \ldots, n$ and the encryption algorithm is additive homomorphic, $D_2(c'_i) = D_2(c^{r_i}_{\pi(i)}) = r_i D_2(c_{\pi(i)}) \bmod p_2$ for $i = 1, 2, \ldots, n$. Suppose $D_2(c_j) = 0$ where $1 \le j \le n$, then $D_2(c'_{\pi^{-1}(j)}) = r_{\pi^{-1}(j)} \times D_2(c_j) = r_{\pi^{-1}(j)} \times 0 \bmod p_2 = 0$. So there is at least one zero in $D_2(c'_1), D_2(c'_2), \ldots, D_2(c'_n)$. Therefore, $ZM(A_1,\ A_2 \mid c_1, c_2, \ldots, c_n) = \mathsf{TRUE}$. □

Theorem 2. *The specialized zero test is sound in the negatively-malicious model. More precisely, if nobody deviates from the protocol and* $ZM(A_1,\ A_2 \mid c_1, c_2, \ldots, c_n) = \mathsf{TRUE}$*, then there is at least one null ciphertext in* $c_1, c_2, \ldots, c_n$*.*

Proof: As $c'_i = c_{\pi(i)}^{r_i}$ for $i = 1, 2, \ldots, n$ and the encryption algorithm is additive homomorphic, $D_2(c'_i) = D_2(c_{\pi(i)}^{r_i}) = r_i D_2(c_{\pi(i)}) \bmod p_2$ for $i = 1, 2, \ldots, n$. As $ZM(A_1,\ A_2 \mid c_1, c_2, \ldots, c_m) = \mathsf{TRUE}$, there is at least one zero encrypted in $c'_1, c'_2, \ldots, c'_n$. Suppose $D_2(c'_j) = 0$ and $1 \le j \le n$. Then $r_j D_2(c_{\pi(j)}) = 0 \bmod p_2$. As p_2 is a prime and r_j is chosen from $Z_{p_2} - \{0\}$, $D_2(c_{\pi(j)}) = 0$. Therefore, there is at least one null ciphertext in $c_1, c_2, \ldots, c_n$. □

Theorem 3. *The specialized zero test is private. More precisely, if* A_1 *and* A_2 *do not collude, the only knowledge of either of them about* $D_2(c_1), D_2(c_2), \ldots, D_2(c_n)$ *is the test result.*

Proof: As A_1 has no knowledge of the private key and the encryption algorithm is semantically-secure, nothing about $D_2(c_1), D_2(c_2), \ldots, D_2(c_n)$ is revealed to him if A_2 does not help to decrypt any message. As A_2 does not collude with A_1, A_2 only tells A_1 the test result, which is A_1's only knowledge about $D_2(c_1), D_2(c_2), \ldots, D_2(c_n)$.

Although A_2 has the private key, his knowledge is limited by the ciphertexts sent to him. As A_1 does not collude with him, only $c'_1, c'_2, \ldots, c'_n$ are sent to A_2. So his only knowledge from the test is $D_2(c'_1)||D_2(c'_2)||\ldots||D_2(c'_n)$, which is called his knowledge transcript. Suppose T_1 and T_2 are two knowledge transcripts from two inputs with the same test result. Note that $c'_i = c_{\pi(i)}^{r_i}$, p_2 is a prime and r_i is randomly chosen from $Z_{p_2} - \{0\}$ as A_1 does not collude with A_2. So $D_2(c'_i)$ is distributed uniformly in $Z_{p_2} - \{0\}$ if $D_2(c_{\pi(i)}) \neq 0$ or $D_2(c'_i) = 0$ if $D_2(c_{\pi(i)}) = 0$. So if A_1 does not collude with A_2, when the test result is TRUE, both T_1 and T_2 are uniformly distributed in $\{\, T \mid T \in \{Z_{p_2}\}^n, T \text{ contains one } 0\}$; when the test result is FALSE, both T_1 and T_2 are uniformly distributed in $(Z_{p_2} - \{0\})^n$. As A_2's knowledge transcripts from any two inputs with the same test result are indistinguishable from each other without A_1's collusion, no information about the input is revealed to A_2 except for the test result without A_1's collusion. □

4 The New Range Test Protocol

In the new range test protocol, given a ciphertext c encrypted in the first encryption system described in Section 2, the tester runs a two-party protocol with the authority to examine whether $D_1(c)$ is in a certain interval range without knowing or revealing $D_1(c)$. In this protocol there is a limitation about the range size: no more than $p_1/5$, which is of the same magnitude as the size of the message space. As p_1 is very large (e.g. 1024 bits long) in any practical encryption algorithm, the range is large enough for normal applications. For simplicity, it is assumed that the range involved in the test is Z_q where $5q \le p_1$. Note that range test in any consecutive integer range in the message space with a size no more

than $p_1/5$ can be easily reduced to a range test in a same-size range starting from zero due to homomorphism of the encryption algorithm. Two range test protocols are designed in this section based on a principle: $m \in Z_q$ if and only if $m\%q = m$, which can be tested by reducing it to multiple simpler tests and repeatedly exploiting homomorphism of the employed encryption algorithms. Firstly, a correct but only partially sound test protocol in the negatively-malicious model — *basic range test* — is described. Then a correct and sound test protocol in the negatively-malicious model, called *precise range test*, is designed based on two basic range tests.

4.1 Basic Range Test

The basic range test is an interactive protocol between two parties: the tester and the authority. The tester is denoted as A_1, who possesses a ciphertext c in the first encryption system. The authority is denoted as A_2, who possesses the private keys of the two encryption systems. The basic range test protocol includes three steps. In the first step, m, the message encrypted in c is randomly shared between A_1 and A_2. Namely, A_1 holds random integer m_1, A_2 holds random integer m_2 such that $m = m_1 + m_2 \bmod p_1$. In the second step, A_2 transmits $E_2(m_2)$ and $E_2(m_2\%q)$ to A_1. In the third step, A_1 and A_2 perform a specialized zero test, during which A_1 provides some randomised and shuffled ciphertexts and A_2 decrypts them. The basic range test is denoted as $BR\ (\ A_1,\ A_2 \mid c\ )$ and described in Figure 2, such that

$$BR\ (\ A_1,\ A_2 \mid c\ ) = \begin{cases} \mathsf{TRUE} & \text{if (3)} = \mathsf{TRUE} \\ \mathsf{FALSE} & \text{if (3)} = \mathsf{FALSE} \end{cases}$$

Theorem 4. *The basic range test is correct in the negatively-malicious model. More precisely, if nobody deviates from the protocol and* $0 \le D_1(c) < q$*, the specialized zero test in Formula (3) outputs* TRUE.

Proof: Suppose $D_1(c) = m$. As $0 \le D_1(c) < q$, $m\%q = m$. There are two important facts.

- As $c = c_1c_2$, $m = m_1 + m_2 \bmod p_1$. So, either (1): $m = m_1 + m_2$ or (2): $m = m_1 + m_2 - p_1$.
- It is always true that either (a): $(m_1 + m_2)\%q = m_1\%q + m_2\%q$ or (b): $(m_1 + m_2)\%q = m_1\%q + m_2\%q - q$.

So the proof is given in four different cases by combining the two possibilities in the first fact, (1) and (2), with the two possibilities in the second fact, (a) and (b): (1a), (1b), (2a) and (2b).

- (1a): According to additive homomorphism of the encryption algorithm

$$D_2(e_1e_2/(c_1'c_2')) = D_2(e_1e_2/(E_2(m_1)E_2(m_2))) = D_2(e_1) + D_2(e_2) - (D_2(E_2(m_1)) + D_2(E_2(m_2)) \bmod p_2 = m_1\%q + m_2\%q - (m_1 + m_2) \bmod p_2$$

1. A_1 randomly chooses m_1 from Z_{p_1}. He calculates $c_1 = E_1(m_1)$ and sends $c_2 = c/c_1$ to A_2.
2. (a) A_2 calculates $m_2 = D_1(c_2)$.
   (b) A_2 calculates $c_2' = E_2(m_2)$ and $e_2 = E_2(m_2 \% q)$ and sends them to A_1.
3. (a) A_1 calculates $c_1' = E_2(m_1)$ and $e_1 = E_2(m_1 \% q)$.
   (b) A_1 needs to perform the following logic test with the help of A_2:

$$D_2(e_1e_2/(c_1'c_2')) = 0 \ \vee \ D_2(e_1e_2/(c_1'c_2'E_2(q))) = 0 \ \vee \quad (2)$$
$$D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q))) = 0 \ \vee \ D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q - q))) = 0 \ \vee$$
$$D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q + q))) = 0$$

In logic expression (2), either all the five clauses are false or only one of them is true. So the logic test of (2) can be implemented through a specialized zero test:

$$ZM(A_1, A_2 \mid e_1e_2/(c_1'c_2'),\ e_1e_2/(c_1'c_2'E_2(q)),\ e_1e_2/(c_1'c_2'E_2(p_1\%q)),$$
$$e_1e_2/(c_1'c_2'E_2(p_1\%q - q)),\ e_1e_2/(c_1'c_2'E_2(p_1\%q + q))) \quad (3)$$

Fig. 2. Basic range test

According to Condition (1) and Condition (a),

$$D_2(e_1e_2/(c_1'c_2')) = (m_1 + m_2)\%q - m \bmod p_2 = m\%q - m \bmod p_2 = 0$$

– similarly to the cases of (1a), in the cases of (1b), (2a) and (2b), $D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q))) = 0$ or $D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q - q))) = 0$ or $D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q + q))) = 0$

In summary, it is always true that

$$D_2(e_1e_2/(c_1'c_2')) = 0 \ \vee \ D_2(e_1e_2/(c_1'c_2'E_2(q))) = 0 \ \vee$$
$$D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q))) = 0 \ \vee \ D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q - q))) = 0$$
$$\vee \ D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q + q))) = 0$$

As $ZM()$ is correct according to Theorem 1,

$$ZM(A_1, A_2 \mid e_1e_2/(c_1'c_2'),\ e_1e_2/(c_1'c_2'E_2(q)),\ e_1e_2/(c_1'c_2'E_2(p_1\%q)),$$
$$e_1e_2/(c_1'c_2'E_2(p_1\%q - q)),\ e_1e_2/(c_1'c_2'E_2(p_1\%q + q))) = \mathsf{TRUE}$$

□

Lemma 1. *If* $\sum_{i=1}^{n}(-1)^{m_i}x_i = 0 \bmod p$ *and* $\sum_{i=1}^{n}|x_i| < p$ *where* $m_i = 0$ *or* 1 *for* $i = 1, 2, \ldots, n$, *then* $\sum_{i=1}^{n}(-1)^{m_i}x_i = 0$.

Proof of Lemma 1 is very simple and is not presented due to space limitation.

Theorem 5. *The basic range test is partially sound in the negatively-malicious model. More precisely, if nobody deviates from the protocol and the specialized zero test in Formula (3) outputs* TRUE, *then* $0 \le D_1(c) < 3q$.

Proof: As $ZM()$ is sound according to Theorem 2

$$D_2(e_1e_2/(c_1'c_2')) = 0 \ \vee \ D_2(e_1e_2/(c_1'c_2'E_2(q))) = 0 \ \vee$$
$$D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q))) = 0 \ \vee \ D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q-q))) = 0$$
$$\vee \ D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q+q))) = 0$$

when

$$ZM(e_1e_2/(c_1'c_2'),\ e_1e_2/(c_1'c_2'E_2(q)),\ e_1e_2/(c_1'c_2'E_2(p_1\%q)),$$
$$e_1e_2/(c_1'c_2'E_2(p_1\%q-q)),\ e_1e_2/(c_1'c_2'E_2(p_1\%q+q))) = \mathsf{TRUE}$$

In the following proof $m_1\%q+m_2\%q$ is calculated with the help of homomorphic property $m_1\%q + m_2\%q = D_2(e_1) + D_2(e_2) = D_2(e_1e_2) \bmod p_2$ and under the condition of every clause in Equation (4). Each clause corresponds to a case in the proof, while each case is divided into two sub-cases: either $m = m_1 + m_2$ or $m = m_1 + m_2 - p_1$.

- If $D_2(e_1e_2/(c_1'c_2')) = 0$, then $D_2(e_1e_2) = D_2(c_1'c_2') = D_2(E_2(m_1)E_2(m_2)) = m_1 + m_2 \bmod p_2$.
  - If $m = m_1 + m_2$, then

    $$m_1\%q + m_2\%q = D_2(e_1e_2) \bmod p_2 = m_1 + m_2 \bmod p_2 = m \bmod p_2$$

    Note that $|m_1\%q|+|m_2\%q|+|m| < 2q+p_1 < p_2$ as $5q \le p_1$ and $p_2 \ge 3p_1$. So according to Lemma 1, $m_1\%q + m_2\%q = m$. Therefore, $m < 2q$.
  - If $m = m_1 + m_2 - p_1$, then

    $$m_1\%q + m_2\%q = D_2(e_1e_2) \bmod p_2 = m_1 + m_2 \bmod p_2 = m + p_1 \bmod p_2$$

    Note that $|m_1\%q| + |m_2\%q| + |m| + |p_1| < 2q + 2p_1 < p_2$ as $5q \le p_1$ and $p_2 \ge 3p_1$. So according to Lemma 1, $m_1\%q + m_2\%q = m + p_1$, which is impossible as $m_1\%q + m_2\%q < 2q < p_1 < m + p_1$. Therefore, it is impossible that $m = m_1 + m_2 - p_1$ when $D_2(e_1e_2/(c_1'c_2')) = 0$.

  So, $m < 2q$.
- If $D_2(e_1e_2/(c_1'c_2'E_2(q))) = 0$, it can be similarly proved that $m < q$.
- If $D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q))) = 0$, it can be similarly proved that $m < 2q$.
- If $D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q-q))) = 0$, it can be similarly proved that $m < 3q$.
- If $D_2(e_1e_2/(c_1'c_2'E_2(p_1\%q+q))) = 0$, it can be similarly proved that $m < q$.

In summary, it is always true that $m < 3q$. □

Theorem 6. *The basic range test is private. More precisely, if A_1 and A_2 do not collude, the only knowledge of either of them about $D_1(c)$ is the test result.*

Proof: A_1's total knowledge from the basic range test about $D_1(c)$ is the test result as the employed encryption algorithms are semantically secure and only A_2 knows the private key. So A_1's only knowledge about $D_1(c)$ in the basic range test is the test result if A_2 does not collude with him.

Without A_1's collusion, A_2's total knowledge about $D_1(c)$ is m_2 and T, which is his knowledge transcript in the special zero test. So A_2's knowledge transcript in the basic range test is $m_2||T$. Theorem 3 illustrates that T reveals no information except for the test result if A_1 does not collude with A_2. If A_1 does not collude with A_2, m_2 is uniformly distributed in Z_{p_1} and independent of $D_1(c)$ or T. So A_2's knowledge transcript in the basic range test reveals no information about $D_1(c)$ except for the range test result if A_1 does not collude with him. Therefore, without A_1's collusion, A_2's only knowledge about $D_1(c)$ in the basic range test is the test result. □

The largest size of the range in the basic range test, q, is of the same magnitude as p_1. The basic range test is efficient and has a constant cost independent of the range size.

4.2 Precise Range Test

As partial soundness limits the application of the basic range test, it is upgraded to precise range test, which is absolutely sound. More precisely, precise range test outputs TRUE if and only if the encrypted message is in the range. The precise range test of a ciphertext c in the first encryption system is denoted as $PR\ (\ A_1,\ A_2 \mid c\ )$, such that $PR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE} \Longleftrightarrow 0 \leq D_1(c) < q$. The precise range test of c is described in Figure 3, in which $PR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE}$ guarantees $0 \leq D_1(c) < 3q$ while $BR\ (\ A_1,\ A_2 \mid E_1(q-1)/c\ ) = \mathsf{TRUE}$ guarantees $D_1(c) \in \{0, 1, \ldots, q-1\} \cup \{p_1 - 2q + 1, p_1 - 2q + 2, \ldots, p_1\}$. The intersection of the two ranges is Z_q.

1. A_1 prepares two basic range tests $BR\ (\ A_1,\ A_2 \mid c\ )$ and $BR\ (\ A_1,\ A_2 \mid E_1(q-1)/c\ )$.
2. A_1 presents the two basic range tests to A_2 in a random order.
3. A_2 finishes the two basic range tests and tells A_1 whether both basic range tests output TRUE and no more information.
4.

$$PR\ (\ A_1,\ A_2 \mid c\ ) = \begin{cases} \mathsf{TRUE} & \text{if } BR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE} \text{ and} \\ & BR\ (\ A_1,\ A_2 \mid E_1(q-1)/c\ ) = \mathsf{TRUE} \\ \mathsf{FALSE} & \text{otherwise} \end{cases} \quad (4)$$

Fig. 3. Precise range test

Theorem 7. *The precise range test is correct in the negatively-malicious model. More precisely, if nobody deviates from the protocol and* $0 \leq D_1(c) < q$*, then* $PR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE}$.

Proof: As $0 \leq D_1(c) < q$, according to Theorem 4, $BR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE}$. As $0 \leq D_1(c) < q$ and the encryption algorithm is additive homomorphic, $D_1(E_1(q-1)/c) = q - 1 - D_1(c) < q$. So according to Theorem 4, $BR\ (\ A_1,\ A_2 \mid (E_1(q-1)/c\ ) = \mathsf{TRUE}$. Therefore, $PR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE}$. □

Theorem 8. *The precise range test is absolutely sound in the negatively-malicious model. More precisely, if nobody deviates from the protocol and* $PR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE}$*, then* $0 \leq D_1(c) < q$*.*

Proof: $BR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE}$ and $BR\ (\ A_1,\ A_2 \mid (E_1(q-1)/c\ ) = \mathsf{TRUE}$ as $PR\ (\ A_1,\ A_2 \mid c\ ) = \mathsf{TRUE}$. So, according to Theorem 5 and additive homomorphism of the encryption algorithm, $0 \leq D_1(c) < 3q$ and $(q - 1 - D_1(c))\%p_1 = D_1(E_1(q-1)/c) < 3q$. The fact $(q - 1 - D_1(c))\%p_1 < 3q$ implies $0 \leq D_1(c) < q$ or $D_1(c) > p_1 - 2q$. As $5q \leq p_1$, the fact $D_1(c) > p_1 - 2q$ implies $D_1(c) \geq 3q$. Therefore, $D_1(c) < 3q \quad \wedge \quad (D_1(c) < q \ \vee \ D_1(c) \geq 3q)$. Namely, $0 \leq D_1(c) < q$. □

As the employed encryption algorithms are semantically secure and A_1 knows no private key, his total knowledge about $D_1(c)$ is the test result if A_2 does not collude with him. So the precise range test is private to A_1. More precisely, if A_2 does not collude with A_1, A_1's only knowledge about $D_1(c)$ is the test result. Note that the precise range test only employs two basic range tests, so it is not completely private to A_2. According to Theorem 6, A_2's only knowledge in the precise range test are the results of the two basic range tests if A_1 does not collude with him. When the precise range test outputs TRUE, A_2's only knowledge is the result of the precise range test without A_1's collusion as the precise range test outputs TRUE if and only if both basic range tests output TRUE. However, when the precise range test outputs FALSE, A_2 knows whether $-2q < D_1(c) < 3q$. If one basic range test outputs FALSE and the other outputs TRUE, A_2 knows that $-2q < D_1(c) < 3q$. Otherwise, A_2 knows that $3q \leq D_1(c) \leq p_1 - 2q$. So, complete privacy is sacrificed in the precise range test to achieve absolute soundness in the negatively-malicious model.

The largest size of the range in the precise range test, q, is of the same magnitude as p_1. The precise range test is efficient and has a constant cost independent of the range size.

5 Conclusion

A range test protocol is proposed, which can correctly and soundly test whether a ciphertext contains a message in a certain interval range without revealing the message. If the tester wants, he can get the correct test result with an overwhelmingly large probability. Unlike the existing related techniques, the new protocol is efficient, accepts large enough range size and does not need a prover with knowledge of the message. If privacy is desired in the actively-malicious model, cut-and-choose mechanism can be employed to satisfy it.

Acknowledgement

The research in this paper was supported by a research grant from NICT, Japan.

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. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Berlin, 2000. Springer-Verlag.
3. J Camenisch and M Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 413–430, Berlin, 1999. Springer-Verlag.
4. A Chan, Y Frankel, and Y Tsiounis. Easy come - easy go divisible cash. 1998. Available as `http://www.ccs.neu.edu/home/yiannis/`.
5. R. Cramer, I. B. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.
6. Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *ICISC 2002*, pages 389–406, 2002.
7. Wenbo Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In *PKC 1998*, volume 1431 of *Lecture Notes in Computer Science*, pages 27–42, Berlin, 1998. Springer.
8. Kazumasa Omote and Atsuko Miyaji. A second-price sealed-bid auction with the discriminant of the p-th root. In *Financial Cryptography 2002*, volume 2357 of *Lecture Notes in Computer Science*, pages 57–71, Berlin, 2002. Springer.
9. 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.
10. Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. An efficient and verifiable solution to the millionaire problem. In *ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 315–330, Berlin, 2004. Springer-Verlag.
11. Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Multiplicative homomorphic e-voting. In *INDOCRYPT 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 61–72, Berlin, 2004. Springer-Verlag.
12. Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Ciphertext comparison, a new solution to the millionaire problem. In *ICICS 2005*, volume 3783 of *Lecture Notes in Computer Science*, pages 84–96, Berlin, 2005. Springer-Verlag.

Efficient Primitives from Exponentiation in $\mathbb{Z}_p$

Shaoquan Jiang

Department of Computer Science,
University of Electronic Science and Technology of China,
ChengDu, China 610054
jiangshq@calliope.uwaterloo.ca

Abstract. Since Diffie-Hellman [12], many secure systems, based on discrete logarithm or Diffie-Hellman assumption in $\mathbb{Z}_p$, were introduced in the literature. In this work, we investigate the possibility to construct efficient primitives from exponentiation techniques over $\mathbb{Z}_p$. Consequently, we propose a new pseudorandom generator, where its security is proven under the decisional Diffie-Hellman assumption. Our generator is the most efficient among all generators from $\mathbb{Z}_p^*$ that are provably secure under standard assumptions. If an appropriate precomputation is allowed, our generator can produce $O(\log \log p)$ bits per modular multiplication. This is the best possible result in the literature (even improved by such a precomputation as well). Interestingly, our generator is the first provably secure under a decisional assumption and might be instructive for discovering potentially more efficient generators in the future. Our second result is a new family of universally collision resistant hash family (CRHF). Our CRHF is provably secure under the discrete log assumption and is more efficient than all previous CRHFs that are provably secure under standard assumptions (especially without a random oracle). This result is important, especially when the unproven hash functions (e.g., MD4, MD5, SHA-1) were broken by Wang et al. [37, 38, 39].

1 Introduction

Diffie-Hellman protocol [12] is an exponentiation based key exchange procedure. It is provably secure (against a passive attack) [8] under a now called *decisional Diffie-Hellman* (DDH) assumption. Since then, Diffie-Hellman techniques in $\mathbb{Z}_p^*$ have been largely employed to construct secure systems, see a very partial list of random examples: key exchange [24, 8], encryption [14, 10], key escow [3]. As provable security is always related to a mathematically hard problem, the systems above are usually proven secure under an assumption of either discrete log, computational Diffie-Hellman, or the decisional Diffie-Hellman. These assumptions, throughout more than twenty years' tests [1, 7, 28, 34, 35], have become widely accepted standard assumptions. Naturally, exponentiation in $\mathbb{Z}_p^*$ has served as the main technique in embedding a hard problem into such a system. In this work, we investigate the possibility to construct efficient and secure primitives from exponentiations in $\mathbb{Z}_p^*$. Consequently, we obtain a new pseudorandom

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 259–270, 2006.
© Springer-Verlag Berlin Heidelberg 2006

generator and an efficient family of collision resistant hash function. Before going on, we first review the research status in these two topics.

Pseudorandom Generator. A key stream generator is a polynomial time algorithm, which upon a short secret outputs a poly-length binary stream. Encryption of a message is to bit-wise XOR it with the underlying key stream. Decryption works in the obvious way. Key stream generators are widely used in the real world, from ancient military communications to today's cell phone applications. The notion of pseudorandom generator (or cryptographically secure key stream generator) was formally defined by Blum and Micali [6] and Yao [40]. In these (different but equivalent) definitions, a generator is said to be secure if no probabilistic polynomial time algorithm can distinguish the generator's output from a uniformly random stream. Blum and Micali [6] constructively showed that a one-way permutation suffices to construct a pseudorandom generator. Then, they showed that a generator that iterates $g^x \pmod p$ for a large prime p and extracts the most significant bit of x, is secure. This is improved by Long and Wigderson [27] for extracting the most $O(\log\log p)$ significant bits in each iteration. Blum, Blum and Shub [5] showed that the parity sequence with an iteration function $x^2 \pmod N$ is secure, where N is a RSA composite [32]. Yao [40] and Goldwasser et al [16] constructed more pseudorandom generators from the intractability of factoring. Alex et al. [2] showed that inverting a RSA function is equivalent to guessing the least significant bit of the input significantly better than 1/2. They further showed that the least significant $O(\log\log N)$ bits of RSA function $x^e \pmod N$ are simultaneously secure. This results was also obtained by Vazirani and Vazirani [36]. This implies a pseudorandom generator with each iteration extracting $O(\log\log N)$ bits. Hastad et al. [21] showed that generally a secure pseudorandom generator exists if and only if a one-way function exists. Hastad et al. [22] showed that the most (or least) significant $\lceil \frac{n}{2} \rceil$ bits of exponentiation function modulus a RSA composite N are simultaneously secure, where $|N| = n$. This results in a HSS generator that extracts half of the RSA input in each iteration. Goldreich and Rosen [20] improved the HSS generator with more efficient computation in each iteration. These generators [22, 20] are further improved by Dedic et al. [13] by removing the requirement of an extra extractor or hash. Recently, Patel et al [29] and Gennaro [15] constructed a very efficient generator that output almost n bits for each iteration, while his construction assumes a non-standard Discrete Logarithm Short Exponent Assumption.

Collision Resistant Hash Function. Roughly speaking, collision resistant hash function (CRHF) is a function for which it is hard to find two inputs with an identical function value. CRHF was first proposed by Damgard [11] from claw-free permutations. Their construction requires $O(1/\log r)$ time (r is a fixed integer) to process one bit while none of the concrete schemes in his paper can achieve $O(1/\log k)$ time per bit, where k is the security parameter. CRHF in Pointcheval [31] and Shamir and Tauman [33] require 1.5 modular multiplication per bit. Goldwasser et al. [17] requires one modular squaring per bit. Bellare et

al. [4] proposed a very efficient CRHF but it assumes random oracle. Efficient CRHF from non-standard assumptions are proposed in Peikert and Rosen [30] and Contini et al. [9]. To our knowledge, no construction, provably secure under a standard assumption, has achieved $O(1/\log k)$ time per bit.

1.1 Contribution

In this work, through manipulating an exponentiation technique in $\mathbb{Z}_p^*$ (p a prime), we construct a new pseudorandom generator and a new family of collision resistant hash function.

Our generator can output one bit per one modular multiplication and more efficient than the previous generators from $\mathbb{Z}_p^*$ (i.e., [6, 27]) that are probably secure under a standard assumption. Our generator is the first one provably secure under a decisional assumption and might be instructive for discovering potentially more efficient generators. But we point out that, comparing with factoring based construction, our generator is asymptotically the same efficient as HSS generator [22] and BBS generator [5] but less efficient than GR generator [20], ACGS generator [2] generator and DRV generator [13]. We stress that here the comparison assumes $|p| = |N|$, where N is the RSA modulus used in factoring based construction. This is justified by the following: (1) in the current start of art results, factoring and discrete log problems have the same heuristic cryptanalytic result [34, 25]; (2) no known cryptanalytic result can separate the decisional Diffie-Hellman and the discrete log problem in $\mathbb{Z}_p^*$, when p is a safe prime. If an appropriate precomputation is allowed, our generator can output $O(\log\log p)$ bits per modular multiplication, which achieves the current best result as GR generaor, DRV generator and ACGS generator (although DR generator and DRV generator requires less precomputation than ours and ACGS generator needs no precomputation at all).

Our CRHF is provably secure under the discrete log assumption. It requires only $O(\frac{1}{\log\log p})$ time per bit and is more efficient than all previous constructions that are provably secure under standard assumptions (especially without a random oracle). This result is very important, especially when unproven hash constructions (e.g., MD4, MD5, SHA-1) were broken recently [37, 38, 39].

2 Notions

In this section, we introduce some notions that will be used in this paper. Denote $\mathbb{Z}$ the set of integers, $\mathbb{R}$ the set of real numbers. We say a function $\nu : \mathbb{Z} \to \mathbb{R}$ is *negligible*, if for any positive polynomial $p(n)$, $\lim_{n\to\infty} \nu(n)p(n) = 0$.

Definition 1. *We say two ensembles $\{X_n\}_n$ and $\{Y_n\}_n$ are* **computationally indistinguishable***, if for any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ and any polynomial $p(n)$, when n large enough,*

$$|\Pr[\mathcal{A}(X_n) = 1] - \Pr[\mathcal{A}(Y_n) = 1]| < 1/p(n).$$

Definition 2. *Let U_l be a random variable uniformly distributed over $\{0,1\}^l$. We say an efficiently computable function $G : \{0,1\}^\kappa \times \mathbb{Z} \to \{0,1\}^*$ is a* **pseudorandom generator***, if for any polynomially bounded integer $l \in \mathbb{Z}$, $G(U_\kappa, l)$ is computationally indistinguishable from U_l.*

In the above, we only consider the case of a binary generator. We can also generalize it to the setting where the generator output is from an arbitrary domain D (instead of $\{0,1\}$ only). In this case, the above definition is modified such that U_t is uniformly random in D^t and G is a function from $D' \times \mathbb{Z}$ to D^* for some domain D'. We call a generator satisfying the modified definition **a pseudorandom number sequence generator**.

A pseudorandom function is a cryptographic approximation of a random function. Loosely speaking, pseudorandom functions are functions that are indistinguishable from random functions.

Definition 3. *Let $\{F_n\}$ be an ensemble of functions, where $F_n : \{0,1\}^* \to \{0,1\}^{l(n)}$ is a random variable uniformly distributed over some set of functions Ω_n, where l is a fixed integer. If Ω_n consists of all possible functions from $\{0,1\}^*$ to $\{0,1\}^{l(n)}$, then $\{F_n\}$ is called a* **random function ensemble***.*

We use $\mathcal{M}^f$ to denote the algorithm $\mathcal{M}$ with oracle access to the function f (i.e., he can adaptively issue a query x, and in return he will receive the function value $f(x)$). We call such an algorithm $\mathcal{M}$ an *oracle machine*.

Definition 4. *Let $\{H_n\}_n$ with $H_n : \{0,1\}^* \to \{0,1\}^{l(n)}$ be a random function ensemble. Assume $\{F_n\}_n$ with $F_n : \{0,1\}^* \to \{0,1\}^{l(n)}$ is an efficiently sampleable and efficiently computable function ensemble. $\{F_n\}_n$ is said to be* **pseudorandom** *if for any PPT oracle machine $\mathcal{M}$,*

$$|\Pr[\mathcal{M}^{F_n}(1^n) = 1] - \Pr[\mathcal{M}^{H_n}(1^n) = 1]| \tag{1}$$

is negligible.

Definition 5. *A family of efficiently computable functions $\{H_s\}_{s\in\{0,1\}^*}$ from $\{0,1\}^*$ to $\{0,1\}^{l(|s|)}$ is said to be* **Collision Resistant** *if for any PPT algorithm $\mathcal{A}$,*

$$\Pr[(x', x) \leftarrow \mathcal{A}(s) \text{ for } s \leftarrow I(1^n) \text{ s.t. } H_s(x') = H_s(x) \ \&\ x' \neq x] \tag{2}$$

is negligible, where $I(1^n)$ is the index generation for the function family $\{H_s\}$, and the probability is taken over internal coin flips in both $I(1^n)$ and $\mathcal{A}$.

3 Our New Pseudorandom Generator

In this section, we will introduce our new pseudorandom generator. Our generator is provably secure under the decisional Diffie-Hellman assumption.

Let $p = 2q + 1$ and q be two large primes. Assume G_q is the subgroup of $\mathbb{Z}_p^*$ of order q and g is a generator of G_q. Let function $|\cdot|_p : G_q \to \mathbb{Z}_q$ be defined as follows:

$$|x|_p = \begin{cases} x & \text{if } 1 \le x < q, \\ p - x & \text{if } q + 2 \le x < p, \\ 0 & \text{otherwise.} \end{cases}$$

Note that G_q is exactly the set of quadratic residues in $\mathbb{Z}_p^*$. In addition, $\left(\frac{q(q+1)}{p}\right) = \left(\frac{1+q^{-1}}{p}\right) = \left(\frac{-1}{p}\right) = -1$ as $p \equiv 3 \pmod 4$. It follows that either $q \in G_q$ or $q+1 \in G_q$ but not both. More precisely, by *Law of Quadratic Reciprocity* [23], we have that $q \in G_q$ if $q \equiv 1 \pmod 4$ and $q+1 \in G_q$ if $q \equiv 3 \pmod 4$. Further notice that $\left(\frac{p-x}{p}\right) = -1$, $\forall x \in G_q$. Therefore, we have that $|\cdot|_p$ is a 1-1 and onto mapping from G_q to $\mathbb{Z}_q$.

Construction 1. We define a number sequence generator NSG as follows. Let $(A_0, A_1) \in G_q \times G_q$ is the initial secret. Starting from $t = 2$, iteratively define $A_t = g^{|A_{t-1}|_p |A_{t-2}|_p}$. Let $A'_t = g^{|A_t|_p}$. The output sequence is $A'_0, A'_1, A'_2, \cdots$.

Now we show that our number sequence generator is secure under the decisional Diffie-Hellman (DDH) assumption.

Theorem 1. *Under the* decisional Diffie-Hellman *assumption, NSG is a pseudorandom number sequence generator.*

Proof. We need to show that $S_t = A'_0, A'_1, \ldots, A'_t$ is indistinguishable from $U_{t+1} \leftarrow G_q^{t+1}$ for any polynomially bounded t. We use a hybrid argument. Let $S_t^{(v)}$ be S_t, except $A'_0, \ldots, A'_v$ being taken uniformly random from G_q. Thus, $S_t^{(t)}$ is uniformly random in G_q^{t+1}. If there exists an adversary $\mathcal{M}$ distinguishing S_t from U_{t+1}, then $\mathcal{M}$ can distinguish $S_t^{(w)}, S_t^{(w+1)}$ for some $w \ge 1$ (Note the distributions of $S_t^{(0)}$ and $S_t^{(1)}$ are identical). Without loss of generality, assume w is known (otherwise, one can guess it correctly with probability $1/t$). We construct a PPT adversary $\mathcal{D}$ to break the DDH assumption. Upon receiving input (α, β, γ), $\mathcal{D}$ takes $A'_0 \leftarrow G_q, \ldots, A'_{w-2} \leftarrow G_q$, and defines $A'_{w-1} = \alpha, A'_w = \beta, A'_{w+1} = g^{|\gamma|_p}$. For $v = w+2, \ldots, t$, iteratively and normally computes $A_v = g^{|A_{v-1}|_p |A_{v-2}|_p}$ and $A'_v = g^{|A_v|_p}$. Note here $A_{w+2} = \beta^{|\gamma|_p}$. Finally, $\mathcal{D}$ feeds $A'_0, \ldots, A'_t$ to machine $\mathcal{M}$, and outputs whatever he does. When (α, β, γ) is a DH tuple, then the input to $\mathcal{M}$ is distributed exactly as $S_t^{(w)}$; if (α, β, γ) is a random tuple, then the input to $\mathcal{M}$ is distributed exactly as $S_t^{(w+1)}$. Thus, $\mathcal{D}$ has a non-negligible advantage, contradiction. ■

We have showed that Construction 1 is a pseudorandom number sequence generator. But in real applications, we are more interested in a binary generator. A naive idea is to encode the output of NSG into a binary form. However, one can easily show that it does not work. In the following, we construct a simple function to convert an NSG sequence into a binary pseudorandom sequence.

Construction 2. Let $A'_0, A'_1, \ldots$ be the output sequence of NSG in Construction 1. Let $L_k(x)$ be the k least significant bits of x, where k is a positive integer.

Define $B_i = L_k(|A'_i|_p)$ for all $i \geq 0$. Then the output stream of the new generator is set to $B_0, B_1, \cdots$. Denote this binary generator by PSG_2.

Theorem 2. *If $k = |q| - \omega(\log |q|)$, then PSG_2 is a pseudorandom generator, where $\omega(x)$ means $\lim_{x\to\infty} \frac{x}{\omega(x)} = 0$.*

Proof. Denote $Z_t = B_0, B_1, \ldots, B_t$. Let U_t be a random variable uniform in $\{0,1\}^{kt}$. We need to show that Z_t is indistinguishable from U_{t+1} for any polynomially bounded t. Consider a random variable $\tilde{Z}_t = \tilde{B}_0, \tilde{B}_1, \ldots, \tilde{B}_t$, where $\tilde{B}_i = L_k(|C_i|_p)$ and $C_i \leftarrow G_q$ for all $i = 0, 1, \ldots, t$. To prove the theorem, it suffices to show that (1) Z_t and $\tilde{Z}_t$ are indistinguishable, and that (2) $\tilde{Z}_t$ and U_{t+1} are indistinguishable.

- *Z_t and $\tilde{Z}_t$ are indistinguishable.* If this is not true, there exists a PPT algorithm $\mathcal{D}_1$ to distinguish them. Then, one can constructs an algorithm $\mathcal{D}'_1$ to distinguish S_t (in Construction 1) from $V_{t+1} \leftarrow G_q^{t+1}$ as follows. $\mathcal{D}'_1$ first applies operator $L_k()$ to the received sequence, then feeds the produced sequence to $\mathcal{D}_1$ and outputs whatever he does. When the input to $\mathcal{D}'_1$ is S_t, then the input to $\mathcal{D}_1$ is distributed exactly as Z_t; otherwise, it is distributed as $\tilde{Z}_t$. Thus, a non-negligible advantage of $\mathcal{D}_1$ implies a non-negligible advantage of $\mathcal{D}'_1$, contradicting Theorem 1.
- *$\tilde{Z}_t$ and U_{t+1} are indistinguishable.* Note that $|\cdot|_p$ is 1-1 and onto mapping from G_q to $\mathbb{Z}_q$. Thus, if C_i is uniform in G_q, $|C_i|_p$ is uniform in $\mathbb{Z}_q$. Let $\tilde{C}_i = |C_i|_p$. Thus, defining $\tilde{B}_i = L_k(|C_i|_p)$ with $C_i \leftarrow G_q$ is equivalent to defining $\tilde{B}_i = L_k(\tilde{C}_i)$ with $\tilde{C}_i \leftarrow \mathbb{Z}_q$. We consider the latter when defining $\tilde{Z}_t$. Consider equation $X \equiv w \pmod{2^k}$ with an unknown X over $\mathbb{Z}_q$, where $w \in \{0, 1, \ldots, 2^k - 1\}$. For any $w \in \{0, 1, \ldots, 2^k - 1\}$, there are either $\lfloor \frac{q}{2^k} \rfloor$ or $\lfloor \frac{q}{2^k} \rfloor + 1$ solutions in $\mathbb{Z}_q$ for X. Thus, $\tilde{B}_i = w$ with probability $2^{-k} + \frac{\delta_w}{q}$ for some $\delta_w \in [-1, 1]$. Thus, the statistic distance between $\tilde{Z}_t$ and U_{t+1} (denoted by $\mathsf{dist}[\tilde{Z}_t, U_{t+1}]$) is at most $\sum_{i=0}^{t} \mathsf{dist}[\tilde{B}_i, U_{t+1}^{(i)}] \leq \frac{(t+1)2^k}{q} \leq \frac{t+1}{2^{\omega(\log |q|)}}$, negligible, where $U_{t+1}^{(i)}$ is the ith k-bit component of U_{t+1}. For any distinguisher $\mathcal{D}_2$, we have

$$
\begin{aligned}
&|\Pr[\mathcal{D}_2(\tilde{Z}_t) = 1] - \Pr[\mathcal{D}_2(U_{t+1}) = 1]| \\
&= \textstyle\sum_{w \in \mathbb{Z}_q^{t+1}} |\Pr[\tilde{Z}_t = w] \Pr[\mathcal{D}_2(w) = 1] - \Pr[\mathcal{D}_2(w) = 1] \Pr[U_{t+1} = w]| \\
&= \textstyle\sum_{w \in \mathbb{Z}_q^{t+1}} \Pr[\mathcal{D}_2(w) = 1] |\Pr[\tilde{Z}_t = w] - \Pr[U_{t+1} = w]| \\
&\leq \textstyle\sum_{w \in \mathbb{Z}_q^{t+1}} |\Pr[\tilde{Z}_t = w] - \Pr[U_{t+1} = w]| \\
&= \mathsf{dist}[\tilde{Z}_t, U_{t+1}],
\end{aligned}
$$

  negligible. ∎

From Theorem 2, we immediately have the following corollary.

Corollary 1. *When $k = |q| - \left\lfloor \frac{|q|}{c} \right\rfloor$ for a constant $c > 1$, the resulting PSG_2 is a cryptographically secure pseudorandom generator.*

Corollary 2. *When $k = |p| - \log^2 |p|$, then the resulting PSG_2 is a pseudorandom generator.*

Remark 1. In each iteration, our generator involves two modular exponentiations. By Lim and Lee [26], one modular exponentiation can be done in $|p|/2$ modular multiplications, assuming pre-computation of $g^{2^i}, i = 0, \ldots, |p|$ and $g^{2^j + 2^{j+[|p|/2]}}, j = 0, \ldots, [|p|/2]$. So our generator can asymptotically output one bit for each multiplication modular a prime p. This result is better than other generators from $\mathbb{Z}_p^*$ [6, 27] that are provably secure under a standard assumption. More interestingly, in each iteration, our generator can output bits of length almost $|p|$, while no previous generator (including factoring based generator) proven secure in the standard assumption, has achieved this. Thus, the construction here might be interesting for motivating more efficient generators in the future. Note if a more complex version of pre-computation in [26] is adopted, then our generator can output $O(\log \log p)$ bits per modular multiplication. This is the best result in the literature. Specifically, it has been achieved by GR generator and DRV generator with even less precomputation than ours, and by ACGS generator [2] with no precomputation at all.

4 A New Family of Collision Resistant Hash Function

In this section, we construct a family of collision resistant hash function. We start with the following construction. This construction is essentially a realization of the framework [11] but waived of the extra requirement of making the input prefix-free. Later we will show how to obtain more efficient constructions.

Construction 3. Let $p = 2q+1$ and q be two large primes, G_q be the subgroup of order q in $\mathbb{Z}_p^*$. Our hash family $\mathcal{H}_1$ is indexed by (g_0, g_1, s), where $g_0, g_1, s \leftarrow G_q$. Let H be a hash function in $\mathcal{H}_1$ with index (g_0, g_1, s). Upon input $x = x_1 x_2 \cdots x_t \in \{0,1\}^*$, $H(x)$ is computed as follows. First set $Y_{t+1} = s$. For $i = t, t-1, \ldots, 1$, iteratively compute $Y_i = g_{x_i}^{|Y_{i+1}|_p}$. Finally, define $H(x) = Y_1$.

Theorem 3. *Assuming discrete log problem in G_q is hard, $\mathcal{H}_1$ is a collision resistant hash family. In addition, if the input is r bits, then assuming a precomputation, one hash requires a cost of at most $r|p|/2$ modular multiplications.*

Proof. The second argument follows from the precomputation of

$$g_j^{2^i}, g_j^{2^{i+\lfloor \frac{|q|}{2} \rfloor}}, g_j^{2^i + 2^{i+\lfloor \frac{|q|}{2} \rfloor}}, j = 0, 1, i = 1, \cdots, \left\lceil \frac{|q|}{2} \right\rceil.$$

We thus concentrate on the first argument. Assume $H(x) = H(x')$ for some binary string $x = x_1 x_2 \cdots x_t$ and $x' = x'_1 x'_2 \cdots x'_l$ s.t. $x \neq x'$, for some integer l, l'. Without loss of generality, assume $l \geq t$. Then, there must exist a unique index j with $0 \leq j \leq t$ such that $x_1 = x'_1, \ldots, x_j = x'_j$ but $x'_{j+1} \neq x'_{j+1}$, where by default $x_{t+1} = \lambda$ (meaning empty). Let Y_i and Y'_i be the intermediate term with index i when computing $H(x)$ and $H(x')$, respectively. Since $|\cdot|_p$ is a 1-1 and onto mapping, we have the following result.

- Case $j < t$: We immediately have $Y_{j+1} = Y'_{j+1}$. However, $x_{j+1} \neq x'_{j+1}$ and $j+1 \leq t$, $g_{x_{j+1}}^{|Y_{j+2}|_p} = g_{x'_{j+1}}^{|Y'_{j+2}|_p}$. Thus, the discrete $\log_{g_0} g_1$ is obtained. If this event happens with non-negligible probability, we can transform the adversary to break the discrete log assumption, contradiction!
- Case $j = t$: In this case, since $x \neq x'$, it follows that $l > t$. Thus, $s = g_{x'_{t+1}}^{|Y'_{t+2}|_p}$. Therefore, we can obtain the discrete log either $\log_{g_1} s$ or $\log_{g_0} s$. If this happens with non-negligible probability, we can easily transform the adversary to break the discrete log assumption. ■

In the following, we present a more efficient construction of collision resistant hash family.

Construction 4. Let $p = 2q + 1$ and q be two large primes, G_q be the subgroup of order q in $\mathbb{Z}_p^*$. $|q| = k + 1$. Our hash family $\mathcal{H}_2$ is indexed by $(g_{00}, g_{01}, g_{10}, g_{11}, \ldots, g_{(k-1)0}, g_{(k-1)1})$, where $g_{ij} \leftarrow G_q$, $i = 0, \ldots, k-1, j = 0, 1$. Let H be the hash function in $\mathcal{H}_2$ with index $\{g_{ij} : 0 \leq i \leq k-1, j = 0, 1\}$. Upon input $x = x_1 x_2 \cdots x_t$ for $x_i \in \{0,1\}^k$ $(i < t)$ and $|x_t| \leq k$, $H(x)$ is computed as follows. Let $x_j = x_{j0} x_{j1} \cdots x_{j(k-1)}$ for $x_{jl} \in \{0,1\}$. For a l-bit string $z = z_0 z_1 \cdots z_{l-1}$ $(l \leq k)$, denote $g_z = \prod_{j=0}^{l-1} g_{jz_j}$. $Y_{t+1} = s$. For $i = t, t-1, \ldots, 1$, iteratively compute $Y_i = g_{x_i}^{|Y_{i+1}|_p}$. Finally, define $H(x) = Y_1$.

Theorem 4. *Under the discrete log assumption, $\mathcal{H}_2$ is a collision resistant hash family. In addition, if the input is r bits, one function evaluation costs no more than $3r$ modular multiplications.*

Proof. The second argument follows from the facts: an r-bit input is uniformly divided into $t = r/k$ segments; for each segment x_i, g_{x_i} is computed in k modular multiplications; each exponentiation in $\mathbb{Z}_p^*$ costs at most $2|p|$ modular multiplications. We thus focus on the first argument. We show that if the conclusion is wrong, we construct an algorithm $\mathcal{S}$ to solve the discrete log problem over G_q. Assume $\mathcal{H}_2$ is broken by an adversary $\mathcal{A}$. Then $\mathcal{S}$ is constructed as follows. Upon input $g, h \in G_q$, take $w_{ij}^0, w_{ij}^1 \leftarrow Z_q$, and define $g_{ij} = g^{w_{ij}^0} h^{w_{ij}^1}$, for $i = 0, 1, \ldots, k-1, j = 0, 1$. $\mathcal{S}$ provides $p, (g_{10}, g_{11}, \ldots, g_{(k-1)0}, g_{(k-1)1})$ to $\mathcal{A}$ and in turn receives a collision pair x, x' $(x \neq x')$ from $\mathcal{A}$. Assume $x = x_1 x_2 \cdots x_t$ and $x' = x'_1 x'_2 \cdots x'_{t'}$ for some $t, t' > 0$. W.L.O.G, assume $t \leq t'$. Let J be the smallest index such that $x_J \neq x'_J$. Let $Y_{t+1} = Y'_{t'+1} = s$, iteratively define $Y_i = g_{x_i}^{|Y_{i+1}|_p}$ and $Y'_j = g_{x'_j}^{|Y'_{j+1}|_p}$. Thus, we have $Y_J = Y'_J$. So

$$\begin{aligned} & g^{|Y_{J+1}|_p \sum_{i:x_{Ji}=1} w_{Ji}^0} h^{|Y_{J+1}|_p \sum_{i:x_{Ji}=1} w_{Ji}^1} \\ = \; & g^{|Y'_{J+1}|_p \sum_{i:x'_{Ji}=1} w_{Ji}^0} h^{|Y'_{J+1}|_p \sum_{i:x'_{Ji}=1} w_{Ji}^1} \end{aligned}$$

If $|Y'_{J+1}|_p \sum_{i:x'_{Ji}=1} w_{Ji}^1 \neq |Y_{J+1}|_p \sum_{i:x_{Ji}=1} w_{Ji}^1 \pmod q$, then discrete log $\log_g h$ can be efficiently obtained from the above relation. On the other hand, we show the probability that this condition is violated for some J is negligible.

Indeed, let $v = \log_g h, z_{ij} = \log_g g_{ij}$. Then given $z_{ij} = w_{ij}^0 + vw_{ij}^1, i = 0, 1, \ldots, k-1, j = 0, 1$. Thus, given $\{g_{ij} : i = 0, 1, \ldots, k-1, j = 0, 1\}$, $\{w_{ij}^1 : i = 0, \ldots, k-1, j = 0, 1\}$ is independent of the view of $\mathcal{A}$. Thus, $|Y'_{J+1}|_p \sum_{i:x'_{Ji}=1} w_{Ji}^1 = |Y_{J+1}|_p \sum_{i:x_{Ji}=1} w_{Ji}^1 \pmod q$ holds for a particular J with probability $1/q$. So the probability that there exists a J violating the condition is no more than k/q. ■

Now we further improve Construction 4 to reduce the computation cost by factor $\log k$ but increase the storage by a factor k.

Construction 5. Denote the new hash family by $\mathcal{H}_3$. The construction is similar to $\mathcal{H}_2$. But $\mathcal{H}_3$ is indexed by $(g_{00}, g_{01}, g_{0(k-1)}, g_{10}, g_{11}, \ldots, g_{(k-1)(k-1)})$, where $g_{ij} \leftarrow G_q$, $i = 0, \ldots, k-1, j = 0, \ldots, k-1$. Let H be the hash function in $\mathcal{H}_3$ with index $\{g_{ij} : 0 \le i \le k-1, 0 \le j \le k-1\}$. Upon input $x = x_1 x_2 \cdots x_t$ for $x_i \in \{0, \ldots, k-1\}^k$ $(i < t)$ and $|x_t| \le k$, $H(x)$ is computed as follows. Let $x_j = x_{j0} x_{j1} \cdots x_{j(k-1)}$ for $x_{jl} \in \{0, \ldots, k-1\}$. For a k-ary string $z = z_0 z_1 \cdots z_{l-1}$ $(l \le k)$, denote $g_z = \prod_{j=0}^{l-1} g_{jz_j}$. $Y_{t+1} = s$. For $i = t, t-1, \ldots, 1$, iteratively compute $Y_i = g_{x_i}^{|Y_{i+1}|_p}$. Finally, define $H(x) = Y_1$.

Theorem 5. *Under the discrete log assumption, $\mathcal{H}_3$ is a collision resistant hash family. In addition, if the input is r bits, one function costs no more than $3r/\log k$ modular multiplications.*

The proof of the theorem is almost identical to Theorem 4. Construction 5 is computationally more efficient than Construction 4 but it requires more storage. Thus, these two constructions have their own merit of existence.

Remark 2. Our CRHF $\mathcal{H}_3$ is the first construction provably secure in the standard assumption and only requires $O(1/\log k)$ time for each bit. Bellare et al. [4] proposed a notion of *incrementality* for CRHF, which means one can evaluate $H(x|y)$ from $H(x)|y$ instead of from the scratch. Our constructions satisfy this property too.

5 Application to Pseudorandom Function

In this section, we unite our pseudorandom generator and collision resistant hash function into a construction of (universal) pseudorandom functions. To do this, we first show that a collision resistant hash function can be used to extend an input-restricted pseudorandom function to a universal pseudorandom function. To be specific, we just state this result in term of an input-restricted GGM construction [19].

Construction 6. Let $G : \mathcal{D} \to \mathcal{D}^2$ be a pseudorandom generator. Assume $G(x) = G_0(x)|G_1(x)$, where $G_i(x) \in \mathcal{D}, i = 0, 1$. H is a collision resistant hash function. A family of function $F : \mathcal{K} \times \{0,1\}^* \leftarrow \mathcal{D}$ is defined as follows. Given a

private index $k \in \mathcal{K}$ and input $x \in \{0,1\}^*$, compute $u_0 u_1 \cdots u_{t-1} = H(x)$, $F_k(x)$ is defined to be $G_{u_{t-1}} \circ G_{u_{t-2}} \circ \cdots \circ G_{u_0}(k)$.

Theorem 6. *Let H be a collision resistant hash function, G is a pseudorandom generator. Then $\mathcal{F}$ is a family of pseudorandom function.*

Proof. Let X be the input queried by adversary. If a collision in X under $H()$ (i.e., $\exists x_1, x_2 \in X$ with $H(x_1) = H(x_2)$) happens with non-negligible probability, then $H()$ is not collision resistant. Otherwise, the proof is identical to the proof of Theorem 3.6.6 in [18]. The details are omitted. ■

Corollary 3. *Let $\mathcal{H}$ be the collision resistant hash family in Construction 5, and G be the pseudorandom generator in Construction 2. Then for l-bit input x, our pseudorandom function can be computed in roughly $(\frac{3l}{\log k} + 2k^2)$ modular multiplications.*

Remark 3. If we do not apply $\mathcal{H}$ to the input first, then the underlying pseudorandom function (by using construction 3.6.13 in [18]) requires more than $2lk$ modular multiplications, inefficient!

6 Conclusion

In this paper, we proposed a new pseudorandom generator and collision resistant hash function by manipulating exponentiation techniques over $\mathbb{Z}_p$. Our hash function is more efficient than all previous constructions that are provably secure under a standard assumption. Our pseudorandom generator is more efficient than those over $\mathbb{Z}_p$ that are provable under a standard assumption. When an appropriate pre-computation is allowed, it achieves the best possible result in the literature (even improved by such a pre-computation as well). Even though, our constructions are not efficient enough for practical applications. Thus, an interesting open problem is to construct practically efficient and provably secure pseudorandom generators and collision resistant hash functions, where the provable security should be obtained under a widely acknowledged hardness assumption (e.g., NP-completeness).

References

1. L. M. Adleman, A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography (Abstract), *FOCS 1979*, pp. 55-60, 1979.
2. W. Alexi, B. Chor, O. Goldreich, and C. Schnorr, RSA/Rabin Bits are 1/2 + 1/poly(log N) Secure, *FOCS 1984*: 449-457.
3. M. Bellare and S. Goldwasser, Verifiable Partial Key Escrow, *ACM CCS'97*, pp. 78-91, 1997.
4. M. Bellare, D. Micciancio, A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost, *Advances in Cryptology-EUROCRYPT 1997*, pp. 163-192, 1997.

5. L. Blum, M. Blum, M. Shub, A Simple Unpredictable Pseudo-Random Number Generator, *SIAM J. Comput.* 15(2): 364-383 (1986).
6. M. Blum, S. Micali, How to Generate Cryptographically Strong Sequences of Pseudo Random Bits, *FOCS 1982*: 112-117.
7. D. Coppersmith, A. M. Odlyzko, and R. Schroeppel, Discrete Logarithms in GF(p), *Algorithmica* 1(1): 1-15 (1986).
8. R. Canetti and H. Krawczyk, analysis of key-exchange protocols and their use for building secure channels, *Advances in Cryptology-EUROCRYPT 2001*, B. Pfitzmann (Ed.), LNCS 2045, Springer-Verlag, pp. 453-474, 2001.
of key exchange and secure channels, *Advances in Cryptology-EUROCRYPT 2002*, L. R. Knudsen (Ed.), LNCS 2332, Springer-Verlag, pp. 337-351, 2002.
signature-based key-exchange protocol, *Advances in Cryptology-CRYPTO 2002*, M. Yung (Ed.), LNCS 2442, Springer-Verlag, pp. 143-161, 2002.
9. S. Contini and A.K. Lenstra and R. Steinfeld, VSH: an Efficient and Provable Collision Resistant Hash Function, *NIST Cryptographic Hash Workshop 2005*, Maryland, USA, 2005.
10. R. Cramer and V. Shoup, a practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack, *Advances in Cryptology-CRYPTO 1998*, H. Krawczyk (Ed.), LNCS 1462, Springer-Verlag, pp. 13-25, 1998.
11. I. Damgard, Collision Free Hash Functions and Public Key Signature Schemes, *EUROCRYPT 1987*: 203-216.
12. W. Diffie and M. E. Hellman, New Directions in Cryptography, *IEEE Transactions on Information Theory*, vol. IT-22, Nov. 1976, pp: 644-654.
13. N. Dedic, L. Reyzin and S. Vadhan, An Improved Pseudorandom Generator Based on Hardness of Factoring, *Security in Communication Networks 2002*, S. Cimato et al. (Eds.), LNCS 2576, Springer-Verlag, pp. 55-73, 2003.
14. T. El Gamal, a public-key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*, Vol. 31, No. 4, pp. 469-472, 1985.
15. R. Gennaro, An Improved Pseudo-random Generator Based on the Discrete Logarithm Problem , *Journal of Cryptology*, 18(2), pp.91-110, Spring 2005. Early version appeared in CRYPTO'2000.
16. S. Goldwasser, S. Micali and P. Tong, Why and how to establish a private code on a public network, *FOCS'82*, pp. 134-144.
17. S. Goldwasser, S. Micali, and R. L. Rivest, A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks, *SIAM J. Comput.*, 17(2): 281-308 (1988).
18. O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, 2001.
19. O. Goldreich, S. Goldwasser amd S. Micali, How to Construct Random Functions, *Journal of the ACM*, Vol 33, No. 4, pp. 792-807, 1986.
20. O. Goldreich, V. Rosen, On the Security of Modular Exponentiation with Application to the Construction of Pseudorandom Generators, *J. Cryptology*, 16(2): 71-93 (2003).
21. J. Hastad, R. Impagliazzo, L. A. Levin, Michael Luby, A Pseudorandom Generator from any One-way Function, *SIAM J. Comput.* 28(4): 1364-1396 (1999). Early verision is in STOC'89.
22. J. Hastad, A. Schrift and A. Shamir, The Discrete Logarithm Modulo a Composite Hides $O(n)$ Bits, *JCSS*, 47: 376-404, 1993.
23. L. Hua, *Introduction to Number Theory*, Berlin: Springer-verlag, 1982.

24. J. Katz, R. Ostrovsky and M. Yung, efficient password-authenticated key exchange using human-memorable passwords, *Advances in Cryptology-EUROCRYPT 2001*, B. Pfitzmann (Ed.), LNCS 2045, Springer-Verlag, pp. 475-494, 2001.
25. A. K. Lenstra and H. W. Lenstra, Jr. (Eds.), the Developement of the Number Field Sieve, LNM 1554, Springer-Verlag, 1993.
26. C. Lim, P. Lee, More Flexible Exponentiation with Precomputation, *Advances in Cryptology-CRYPTO 1994*, Y. Desmedt (Ed.), LNCS 839, Springer-Verlag, pp. 95-107, 1994.
27. D. L. Long, A. Wigderson, How Discreet is the Discrete Log, *STOC 1983*, pp. 413-420, 1983.
28. A. M. Odlyzko, Discrete Logarithms: The Past and the Future, *Des. Codes Cryptography* 19(2/3): 129-145 (2000).
29. S. Patel and G. S. Sundaram, An Efficient Discrete Log Pseudo Random Generator, *Advances in Cryptology-CRYPTO 1998*, H. Krawczyk (Ed.), LNCS 1462, Springer-Verlag, pp. 304-317, 1998.
30. C. Peikert and A. Rosen, Efficient Collision-Resistant Hashing From Worst-Case Assumptions on Cyclic Lattices, *TCC 2006*.
31. D. Pointcheval, The Composite Discrete Logarithm and Secure Authentication, *Public Key Cryptography 2000*, H. Imai and Y. Zheng (Eds.), LNCS 1751, Springer-Verlag, pp. 113-128, 2000.
32. R. Rivest, A. Shamir and L. Adleman, A Method for Obtaining Digital Signatures and Public-key Cryptosystems, *Communications of ACM*, Vol. 2, pp. 120-126, February 1978.
33. A. Shamir and Y. Tauman, Improved Online/Offline Signature Schemes, *Advances in Cryptology-CRYPTO 2001*, J. Kilian (Ed.), LNCS 2139, Springer-Verlag, pp. 355-367, 2001.
34. O. Schirokauer, Discrete Logarithm and Local Units, *Philosophical Transactions: Physical Science and Engineering*, Vol. 345, No. 1676, pp. 409-423, 1993.
35. Victor Shoup: Lower Bounds for Discrete Logarithms and Related Problems, *Advances in Cryptology-EUROCRYPT 1997*, W. Fumy (Ed.), LNCS 1233, Springer-Verlag, pp. 256-266, 1997.
36. U. Vazirani and V. Vazirani, Efficient and Secure Pseudo-random number generation, *FOCS'84*, pp. 458-463.
37. X. Wang, X. Lai, D. Feng, H. Chen and X. Yu, Cryptanalysis of the Hash Functions MD4 and RIPEMD, *Advances in Cryptology-EUROCRYPT 2005*, R. Cramer (Ed.), LNCS 3494, Springer-Verlag, pp. 1-18, 2005.
38. X. Wang and H. Yu, How to Break MD5 and Other Hash Functions, *Advances in Cryptology-EUROCRYPT 2005*, R. Cramer (Ed.), LNCS 3494, Springer-Verlag, pp. 19-35, 2005.
39. X. Wang, Y. L. Yin and H. Yu, Finding Collisions in Full SHA-1, *Advances in Cryptology-CRYPTO 2005*, V. Shoup (Ed.), LNCS 3621, Springer-Verlag, pp. 17-36, 2005.
40. A. Yao, Theory and Applications of Trapdoor Functions (Extended Abstract), *FOCS 1982*: 80-91.

PA in the Two-Key Setting and a Generic Conversion for Encryption with Anonymity

Ryotaro Hayashi and Keisuke Tanaka

Dept. of Mathematical and Computing Sciences,
Tokyo Institute of Technology,
2-12-1 Ookayama, Meguro-ku,
Tokyo 152-8552, Japan
{hayashi9, keisuke}@is.titech.ac.jp

Abstract. We propose the notion of plaintext awareness in the two-key setting, called PATK. We also prove that if a public-key encryption scheme is secure in the sense of PATK, then it is also secure in the sense of IK-CCA. Since it looks much easier to prove that a public-key encryption scheme is secure in the sense of PATK than to prove directly that it is secure in the sense of IK-CCA, the notion of PATK is useful to prove the anonymity property of public-key encryption schemes.

We also propose the first generic conversion for the anonymity, that is, we prove that the public-key encryption scheme derived from the Fujisaki-Okamoto conversion scheme, where the basic public-key encryption scheme is secure in the sense of IK-CPA, is secure in the sense of IK-CCA in the random oracle model.

Keywords: anonymity, key-privacy, encryption, plaintext awareness in the two-key setting.

1 Introduction

1.1 Background

The classical security requirement of public-key encryption schemes is that it provides privacy of the encrypted data. Popular formalizations such as indistinguishability (IND) or non-malleability (NM), under either the chosen plaintext attack (CPA) or the adaptive chosen ciphertext attack (CCA) are directed at capturing various data-privacy requirements.

The widely admitted appropriate security level for public-key encryption is the indistinguishability against the adaptive chosen ciphertext attack (IND-CCA). A promising way to construct such a public-key encryption scheme is to convert it from primitives which are secure in a weaker sense such as one-wayness (OW), IND-CPA, etc.

Bellare and Rogaway [1] proposed a generic and simple conversion scheme from a one-way trapdoor permutation into a public-key encryption scheme. The scheme created in this way is called OAEP. Fujisaki, Okamoto, Pointcheval, and

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 271–282, 2006.
© Springer-Verlag Berlin Heidelberg 2006

Stern [2] proved that OAEP with a partial one-way trapdoor permutation is secure in the sense of IND-CCA. The OAEP conversion has several variants, such as SAEP, OAEP+, etc.

Fujisaki and Okamoto [3] proposed a simple conversion scheme from weak public-key and symmetric-key encryption schemes into a public-key encryption scheme which is secure in the sense of IND-CCA. This scheme was used to construct the identity-based encryption scheme proposed by Boneh and Franklin [4].

Recently, many conversion schemes which depend on Gap problem, such as REACT and GEM, were proposed.

The public-key encryption schemes derived from the conversion schemes described above meet not only IND-CCA, but also the notion of plaintext awareness (PA). The notion of PA is first proposed by Bellare and Rogaway [1] and refined by Bellare, Desai, Pointcheval, and Rogaway [5] which is, roughly speaking, that nobody can produce a *new* ciphertext without knowing the plaintext. We say that a public-key encryption scheme is secure in the sense of PA if it is secure in the sense of IND-CPA and there exists a knowledge extractor which is a formalization of the above property. In [5], they proved that PA implies IND-CCA. Since it looks much easier to prove that a public-key encryption scheme is secure in the sense of PA than to prove directly it is secure in the sense of IND-CCA, the notion of PA is useful to prove the security of public-key encryption schemes.

Recently, Bellare and Palacio [6] discussed the problem of defining the notion of plaintext-awareness without random oracles and of achieving its concrete schemes.

On the other hand, the notion of PA might be too strong. The schemes described above get a redundant construction. In [7, 8], the conversion schemes without redundancy were proposed. They are secure in the sense of IND-CCA, but does not meet PA. Fujisaki [9] introduced another security notion, called plaintext simulatability (PS). It implies IND-CCA, similar to PA, however, it is a properly weaker notion than PA.

In 2001, Bellare, Boldyreva, Desai, and Pointcheval [10] proposed a new security requirement of encryption schemes called "key-privacy" or "anonymity." It asks that an encryption scheme provides (in addition to privacy of the data being encrypted) privacy of the key under which the encryption was performed. That is, if an encryption scheme provides the key-privacy, then the receiver is anonymous from the point of view of the adversary. They formalized the property of anonymity. This can be considered under either the chosen plaintext attack or the adaptive chosen ciphertext attack, yielding two notions of security, IK-CPA and IK-CCA. (IK means "indistinguishability of keys.").

In addition to the notion of key-privacy, they provided the RSA-based anonymous encryption scheme, RSA-RAEP, which is a variant of RSA-OAEP (Bellare and Rogaway [1], Fujisaki, Okamoto, Pointcheval, and Stern [2]). Recently, Hayashi, Okamoto, and Tanaka [11] proposed the RSA-based anonymous encryption scheme by using the RSACD function. Hayashi and Tanaka [12] constructed the RSA-based anonymous encryption scheme by using the sampling twice technique.

1.2 Our Contribution

In this paper, we propose the notion of plaintext awareness in the two-key setting, called PATK. We say that the public-key encryption scheme Π is secure in the sense of PATK if Π is secure in the sense of IK-CPA and there exists a knowledge extractor for PATK. There are some differences between the definition of a knowledge extractor for PA in [5] and that for PATK (See Section 4). We can see that if there exists a knowledge extractor K for PATK of Π, then we can use K as a knowledge extractor for PA of Π. That is, if the public-key encryption scheme Π is secure in the sense of PATK and IND-CPA, then Π is secure in the sense of PA. However, it is not clear that we can use the knowledge extractor for PA of Π as that for PATK of Π.

We also prove that if a public-key encryption scheme is secure in the sense of PATK, then it is also secure in the sense of IK-CCA. Since it looks much easier to prove that a public-key encryption scheme is secure in the sense of PATK than to prove directly that it is secure in the sense of IK-CCA, the notion of PATK is useful to prove the anonymity property of public-key encryption schemes.

We also propose the first generic conversion scheme for the anonymity from IK-CPA to IK-CCA. We employ the Fujisaki-Okamoto conversion scheme [3]. The public-key encryption scheme derived from their conversion scheme is secure in the sense of IND-CCA in the random oracle model when it consists of a public-key encryption scheme Π^{pub} and a symmetric-key encryption scheme Π^{sym} where

- Π^{pub} is γ-uniform ($\gamma < 1$) and secure in the sense of OW, and
- Π^{sym} is secure in the sense of find-guess (FG).

We prove that the scheme derived from the Fujisaki-Okamoto conversion scheme with the above two and the following two assumptions is secure in the sense of IK-CCA in the random oracle model.

- In Π^{pub}, the message space and the randomness space are common to each user (each public-key).
- Π^{pub} is secure in the sense of IK-CPA.

We can get the public-key encryption scheme which is secure in the sense of IND-CCA and IK-CCA if we assume the above four conditions.

The organization of this paper is as follows. In Section 2, we review the definitions of public-key encryption and symmetric-key encryption. In Section 3 we review the security definitions for public-key encryption and symmetric-key encryption. In Section 4, we propose the notion of plaintext awareness in the two-key setting (PATK), and prove that PATK implies IK-CCA. In Section 5, we review the conversion scheme to IND-CCA proposed by Fujisaki and Okamoto [3]. In Section 6, we propose a generic conversion scheme for the anonymity. More precisely, we prove that the public-key encryption scheme derived from the Fujisaki-Okamoto conversion scheme, where the basic public-key encryption scheme is secure in the sense of IK-CPA, is secure in the sense of IK-CCA in the random oracle model. We conclude in Section 7.

Due to lack of space, we omit proofs in Section 6. See the full version [13] of this paper.

2 Preliminaries

2.1 Public-Key Encryption

In this section, we review the definition of public-key encryption schemes.

In this paper, we mainly consider the anonymity property of encryption schemes proposed in [10]. It asks that the encryption provide (in addition to privacy of the data being encrypted) privacy of the key under which the encryption was performed. In a heterogeneous public-key environment, encryption will probably fail to be anonymous for trivial reasons. For example, different users might be using different cryptosystems, or, if the same cryptosystem, have keys of different lengths. To avoid this problem, we employ some common parameter called *common key* in the definition of encryption schemes, similar to that in [10]. Then, the public key pk includes the corresponding common key I and other information for each user.

Definition 1. *A public-key encryption scheme with common-key generation* $\Pi = (\mathcal{G}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *consists of four algorithms.*

- *The common-key generation algorithm* $\mathcal{G}(1^k)$ *takes as input a security parameter* 1^k *and returns some common key* I*.*
- *The key generation algorithm* $\mathcal{K}(I)$ *is a randomized algorithm that takes as input a common key* I *and returns a pair* (pk, sk) *of keys, a public key and a matching secret key. For given* pk*, the message space* $\mathtt{MSPC}(pk)$ *and the randomness space* $\mathtt{COINS}(pk)$ *of* Π *are uniquely determined.*
- *The encryption algorithm* $\mathcal{E}_{pk}(m; r)$ *is a randomized algorithm that takes a public key* pk *and a plaintext* $m \in \mathtt{MSPC}(pk)$*, and returns a ciphertext* c*, using random coin* $r \in \mathtt{COINS}(pk)$*.*
- *The decryption algorithm* $\mathcal{D}_{sk}(c)$ *is a deterministic algorithm that takes a secret key* sk *and a ciphertext* c*, and returns the corresponding plaintext* m *or a special symbol* $\perp$ *to indicate that the ciphertext* c *is invalid.*

We require that, for any $k \in \mathbb{N}$*, if* $I \leftarrow \mathcal{G}(1^k)$*,* $(pk, sk) \leftarrow \mathcal{K}(I)$*,* $m \in \mathtt{MSPC}(pk)$*, and* $c \leftarrow \mathcal{E}_{pk}(m)$*, then* $m = \mathcal{D}_{sk}(c)$*.*

2.2 Symmetric-Key Encryption

In this section, we review the definition of symmetric-key encryption schemes.

Definition 2. *A symmetric-key encryption scheme* $\Pi = (\mathcal{E}, \mathcal{D})$ *consists of two algorithms.*

- *The encryption algorithm* $\mathcal{E}_x(m)$ *is a deterministic algorithm that takes a symmetric-key* $x \in \mathtt{KSPC}(k)$ *and a message* $m \in \mathtt{MSPC}(k)$*, and returns a ciphertext* c*. Note that* $\mathtt{KSPC}(k)$ *and* $\mathtt{MSPC}(k)$ *are the key space and the message space for* k*, respectively. They are uniquely determined by a security parameter* 1^k*.*

– *The decryption algorithm $\mathcal{D}_x(c)$ is a deterministic algorithm that takes a symmetric key x and a ciphertext c, and returns the corresponding plaintext m.*

We require that, for any $k \in \mathbb{N}$, if $x \in \mathtt{KSPC}(k)$, $m \in \mathtt{MSPC}(k)$, and $c \leftarrow \mathcal{E}_x(m)$, then $m = \mathcal{D}_x(c)$.

3 Security Definitions

In this section, we review the security definitions for public-key encryption and symmetric-key encryption schemes.

3.1 Public-Key Encryption

γ-uniformity. We review a property of public-key encryption, called γ-uniformity, following [3].

Definition 3 (γ-uniformity). *Let $\Pi = (\mathcal{G}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. We say that Π is γ-uniform, if, for any $I \leftarrow \mathcal{G}(1^k)$, $(pk, sk) \leftarrow K(I)$, $m \in \mathtt{MSPC}(pk)$, and $y \in \{0,1\}^*$, $\Pr[r \xleftarrow{R} \mathtt{COINS}(pk) \ : \ y = \mathcal{E}_{pk}(x; r)] < \gamma$.*

One-Wayness. We review a weak security notion for public-key encryption, called one-wayness, following [3].

Definition 4 (OW). *Let $\Pi = (\mathcal{G}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let A be an adversary. We define the advantage of A via* $\mathbf{Adv}^{\mathrm{ow}}_{\Pi,A}(k) =$

$$\Pr[I \leftarrow \mathcal{G}(1^k);\ (pk, sk) \leftarrow \mathcal{K}(I);\ m \xleftarrow{R} \mathtt{MSPC}(pk);\ c \leftarrow \mathcal{E}_{pk}(m) \ : \ A(c, pk) = m].$$

We say that A is a (t, ϵ)-adversary for Π in the sense of OW if A runs in at most time t and archives $\mathbf{Adv}^{\mathrm{ow}}_{\Pi,A}(k) \geq \epsilon$. We say that Π is (t, ϵ)-secure in the sense of OW if there is no (t, ϵ)-adversary for Π in that sense.

Anonymity. In [10], Bellare, Boldyreva, Desai, and Pointcheval formalized the property of "anonymity." This can be considered under either the chosen plaintext attack or the adaptive chosen ciphertext attack, yielding two notions of security, IK-CPA and IK-CCA. (IK means "indistinguishability of keys.") We describe the definition of the anonymity, following [10].

Definition 5 (IK-CPA, IK-CCA [10]). *Let $\Pi = (\mathcal{G}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let A_{cpa} and A_{cca} be adversaries that run in two stages,* find *and* guess*. The adversaries A_{cpa} and A_{cca} have access to some oracles $\mathcal{O}_{\mathrm{cpa}}$ and O_{cca}, respectively. For atk $\in$ {cpa, cca}, we define the advantages of A_{atk} via*

$$\begin{aligned}\mathbf{Adv}^{\text{ik-atk}}_{\Pi,A_{\mathrm{atk}}}(k) = 2 \cdot \Pr[&I \leftarrow \mathcal{G}(1^k);\ (pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathcal{K}(I);\\ &(m, \mathsf{si}) \leftarrow A^{\mathcal{O}_{\mathrm{atk}}}_{\mathrm{atk}}(\mathsf{find}, pk_0, pk_1);\ b \xleftarrow{R} \{0,1\};\ c \leftarrow \mathcal{E}_{pk_b}(m)\\ &: \ A^{\mathcal{O}_{\mathrm{atk}}}_{\mathrm{atk}}(\mathsf{guess}, c, \mathsf{si}) = b] - 1\end{aligned}$$

where $\mathcal{O}_{\mathrm{cpa}} = \epsilon$ *and* $\mathcal{O}_{\mathrm{cca}} = (\mathcal{D}_{sk_0}, \mathcal{D}_{sk_1})$. *Note that* si *is the state information. It contains the public keys* pk_0, pk_1, *the message* m, *and so on. We require that* $m \in \mathtt{MSPC}(pk_0) \cap \mathtt{MSPC}(pk_1)$. *We also require that* A_{cca} *never queries the challenge* c *to either* $\mathcal{D}_{sk_0}$ *or* $\mathcal{D}_{sk_1}$ *in the guess stage.*

We say that A_{cpa} *is a* (t, ϵ)*-adversary for* Π *in the sense of IK-CPA if* A_{cpa} *runs in at most time* t *and achieves* $\mathbf{Adv}^{\text{ik-cpa}}_{\Pi, A_{\mathrm{cpa}}}(k) \geq \epsilon$.

Similarly, we say that A_{cca} *is a* (t, q_d, ϵ)*-adversary for* Π *in the sense of IK-CCA if* A_{cca} *runs in at most time* t, *makes a total number of* q_d *queries to decryption oracles* $\mathcal{D}_{sk_0}$ *and* $\mathcal{D}_{sk_1}$, *and achieves* $\mathbf{Adv}^{\text{ik-cca}}_{\Pi, A_{\mathrm{cca}}}(k) \geq \epsilon$.

We say that Π *is* (t, ϵ)*-secure (respectively* (t, q_d, ϵ)*-secure) in the sense of IK-CPA (resp. IK-CCA) if there is no* (t, ϵ)*-adversary (resp.* (t, q_d, ϵ)*-adversary) for* Π *in the corresponding sense.*

Anonymity in the Random Oracle Model. We can consider the definition of the anonymity in the random oracle model in a similar way as that in the standard model described above.

We define Ω as the map family from an appropriate range. The domain and range depend on the underlying encryption scheme. Even if we choose two random functions that have distinct domains and distinct ranges respectively, we just write the experiment, for convenience, as $G, H \leftarrow \Omega$, instead of preparing two map families.

In the random oracle model, we begin the experiment of A_{atk} described above (which defines advantage) by $H \leftarrow \Omega$. Then, we add the random oracle H to both $\mathcal{O}_{\mathrm{cpa}}$ and $\mathcal{O}_{\mathrm{cca}}$, and allow that for $i \in \{0, 1\}$, $\mathcal{E}_{pk_i}$ and $\mathcal{D}_{sk_i}$ may depend on H (which we write $\mathcal{E}^H_{pk_i}$ and $\mathcal{D}^H_{sk_i}$, respectively).

We define the adversaries in a similar way as those in the standard model, that is, we define a (t, q_h, ϵ)-adversary in the sense of IK-CPA in the random oracle model and a (t, q_h, q_d, ϵ)-adversary in the sense of IK-CCA in the random oracle model where the adversary makes at most q_h queries to H.

We say that Π is (t, q_h, ϵ)-secure (respectively (t, q_h, q_d, ϵ)-secure) in the sense of IK-CPA (resp. IK-CCA) in the random oracle model if there is no (t, q_h, ϵ)-adversary (resp. (t, q_h, q_d, ϵ)-adversary) for Π in the corresponding sense in the random oracle model.

3.2 Symmetric-Key Encryption

Find-Guess. We review a security notion for symmetric-key encryption, called find-guess (FG), following [3].

Definition 6 (FG). *Let* $\Pi = (\mathcal{E}, \mathcal{D})$ *be a symmetric-key encryption scheme. Let* A *be an adversary that runs in two stages,* find *and* guess. *We define the advantage of* A *via*

$$\mathbf{Adv}^{\mathrm{fg}}_{\Pi}(k) = 2 \cdot \Pr[x \xleftarrow{R} \mathtt{KSPC}(k);\ (m_0, m_1, \mathsf{si}) \leftarrow A(\mathsf{find}, k); \\ b \xleftarrow{R} \{0,1\};\ c \leftarrow \mathcal{E}_x(m_b)\ :\ A(\mathsf{guess}, c, \mathsf{si}) = b] - 1.$$

We require that $m_0 \neq m_1$ *and* $m_0, m_1 \in \mathtt{MSPC}(k)$.

We say that A is a (t, ϵ)-adversary for Π in the sense of FG if A runs in at most time t and achieves $\mathbf{Adv}^{\mathrm{fg}}_{\Pi,A}(k) \geq \epsilon$.

We say that Π is (t, ϵ)-secure in the sense of FG if there is no (t, ϵ)-adversary for Π in the sense of FG.

4 Plaintext Awareness in the Two-Key Setting

In this section, we propose the notion of plaintext awareness in the two-key setting (PATK), and prove that PATK implies IK-CCA.

We describe the definition of plaintext awareness in the two-key setting.

Definition 7 (Plaintext Awareness in the two-key setting and Knowledge Extractor for PATK). *Let $\Pi = (\mathcal{G}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let B and K be algorithms, called an adversary for PATK and a knowledge extractor for PATK, respectively. They work in the random oracle model as follows:*

- *B is a (q_h, q_e)-adversary for PATK that takes two public-keys pk_0, pk_1 and an index $i \in \{0, 1\}$, and makes at most q_h queries to H and q_e queries to the encryption oracles, $\mathcal{E}^H_{pk_0}$ and $\mathcal{E}^H_{pk_1}$. B finally outputs $c \notin C$, where*
  - *T_H denotes the set of all pairs of a B's query and the corresponding answer from H, and*
  - *C denotes the set of all answers from $\mathcal{E}^H_{pk_0}$ and $\mathcal{E}^H_{pk_1}$. (Note that C does not contain an information of which encryption oracle responded.)*

  We write this experiment as $(T_H, C, c, pk_i) \leftarrow \mathtt{run}\ B^{H, \mathcal{E}^H_{pk_0}, \mathcal{E}^H_{pk_1}}(pk_0, pk_1, i)$.
- *Knowledge extractor K for PATK takes (T_H, C, c, pk_i) and outputs a string m.*

For any $k \in \mathbb{N}$ and $i \in \{0, 1\}$, we define

$$\mathbf{Succ}^{\mathrm{patk}}_{K,B,\Pi,i}(k) = \Pr[H \leftarrow \Omega;\ I \leftarrow \mathcal{G}(1^k);\ (pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathcal{K}(I);\ (T_H, C, c, pk_i) \leftarrow \mathtt{run}\ B^{H, \mathcal{E}^H_{pk_0}, \mathcal{E}^H_{pk_1}}(pk_0, pk_1, i)\ :\ K(T_H, C, c, pk_i) = \mathcal{D}^H_{sk_i}(c)].$$

We say that K is a $(t_{\mathrm{KETK}}, \lambda, q_h, q_e)$-knowledge extractor for PATK of Π if for any (q_h, q_e)-adversary $B$and $i \in \{0, 1\}$, K runs in at most time t_{KETK} and achieves $\mathbf{Succ}^{\mathrm{patk}}_{K,B,\Pi,i}(k) \geq \lambda$.

We say that Π is $(t_{\mathrm{cpa}}, t_{\mathrm{KETK}}, q_h, q_e, \epsilon, \lambda)$-secure in the sense of PATK if Π is $(t_{\mathrm{cpa}}, q_h, \epsilon)$-secure in the sense of IK-CPA, and there exists a $(t_{\mathrm{KETK}}, \lambda, q_h, q_e)$-knowledge extractor K for PATK of Π.

There are some differences between the definition of PA in [5] and that of PATK. First, the adversary B in our definition receives two public keys and two encryption oracles, while the adversary in the definition of PA receives one public key and one encryption oracle. Second, we define the success probability of B for any index $i \in \{0, 1\}$. This indicates under which key, pk_0 or pk_1, the knowledge extractor K for PATK should decrypt c. Third, in the definition of PA,

the list C contains the answers (ciphertexts) from only one encryption oracle $\mathcal{E}^H_{pk}$. When we prove that PA implies IND-CCA, C plays an important role, that is, C contains the challenge ciphertext of IND-CCA game to give it to the adversary B for PA. In our definition, if we use C to prove that PATK implies IK-CCA, C has to contain the challenge ciphertext of IK-CCA game and the challenge ciphertext is encrypted by either pk_0 or pk_1. Therefore, in our definition, we define that the list C consists of the answers (ciphertexts) from *both* $\mathcal{E}^H_{pk_0}$ *and* $\mathcal{E}^H_{pk_1}$.

It is easy to see that if there exists a knowledge extractor K for PATK of Π, then we can use K as a knowledge extractor for PA of Π. That is, if the public-key encryption scheme Π is secure in the sense of PATK and IND-CPA, then Π is secure in the sense of PA. However, it is not clear that we can use the knowledge extractor for PA of Π as that for PATK of Π. The difficulty of proving this seems to depend on the third difference described above.

We prove the following theorem.

Theorem 1. *If the public encryption scheme Π is $(t_{\mathrm{cpa}}, t_{\mathrm{KETK}}, q_h, 1, \epsilon, \lambda)$-secure in the sense of PATK, then Π is $(t_{\mathrm{cca}}, q_h, q_d, \epsilon')$-secure in the sense of IK-CCA where*

$$t_{\mathrm{cca}} = t_{\mathrm{cpa}} - q_d \cdot t_{\mathrm{KETK}} \text{ and } \epsilon' = \epsilon + 2q_d \cdot (1 - \lambda).$$

Proof. In [5], Bellare, Desai, Pointcheval, and Rogaway proved that PA implies IND-CCA. We prove Theorem 1 in a similar way.

Let A_{cca} be an $(t_{\mathrm{cca}}, q_h, q_d, \epsilon)$-adversary of Π in the sense of IK-CCA. We construct an adversary A_{cpa} of Π in the sense of IK-CPA by using A_{cca}.

We construct the algorithm A_{cpa} as follows. Note that A_{cpa} simulates A_{cca}'s oracles H, $\mathcal{D}_{sk_0}$, and $\mathcal{D}_{sk_1}$ as described below.

1. A_{cpa} initializes two lists, T_H and C to empty.
2. $A_{\mathrm{cpa}}(\mathsf{find}, pk_0, pk_1)$ runs A_{cca} as $(m, \mathsf{si}) \leftarrow A_{\mathrm{cca}}(\mathsf{find}, pk_0, pk_1)$ and outputs (m, si).
3. A_{cpa} receives a challenge ciphertext $\hat{c} = \mathcal{E}^H_{pk_b}(m)$ where $b \overset{R}{\leftarrow} \{0, 1\}$.
4. $A_{\mathrm{cpa}}(\mathsf{guess}, \hat{c})$ runs A_{cca} as $d \leftarrow A_{\mathrm{cca}}(\mathsf{guess}, \hat{c})$ and outputs d.

A_{cpa} simulates A_{cca}'s oracle as follows:

- When A_{cca} makes a query h to H, A_{cpa} makes a query h to *its* oracle H and obtains an answer $H(h)$. Then, A_{cpa} returns $H(h)$ to A_{cca} and puts $(h, H(h))$ into the list T_H.
- When A_{cca} makes a decryption query c to $\mathcal{D}^H_{sk_i}$, A_{cpa} runs the knowledge extractor K as follows.
  - In the find stage, A_{cpa} runs K as $m \leftarrow K(T_H, \epsilon, c, pk_i)$ and returns m to A_{cca}.
  - In the guess stage, A_{cpa} runs K as $m \leftarrow K(T_H, \hat{c}, c, pk_i)$ and returns m to A_{cca}.

To guarantee that the knowledge extractor K for PATK outputs a correct answer (a corresponding plaintext m or an invalid symbol $\perp$), for $j \in \{1, 2, \cdots, q_d\}$ we construct the adversary B_j for PATK as follows. Note that B_j simulates A_{cca}'s oracles H, $\mathcal{D}_{sk_0}$, and $\mathcal{D}_{sk_1}$ as described below. Note that $B_j(pk_0, pk_1, i)$ returns some value and halts when A_{cca} makes its j-th decryption query.

1. B_j initializes two lists, T_H and C to empty.
2. B_j runs A_{cca} as $(m, \mathsf{si}) \leftarrow A_{\mathrm{cca}}(\mathsf{find}, pk_0, pk_1)$.
3. B_j picks a random bit $b \xleftarrow{R} \{0, 1\}$ and makes an oracle query as $\hat{c} \leftarrow \mathcal{E}^H_{pk_b}(m)$.
4. B_j runs $A_{\mathrm{cca}}(\mathsf{guess}, \hat{c})$. (Note that B_j is sure to halt before A_{cca} outputs d. See below.).

$B_j(pk_0, pk_1, i)$ simulates A_{cca}'s oracle as follows:

- When A_{cca} makes a query h to H, A_{cpa} makes a query h to *its* oracle H and obtains an answer $H(h)$. Then, A_{cpa} returns $H(h)$ to A_{cca} and puts $(h, H(h))$ into the list T_H.
- When A_{cca} makes a j'-th decryption query c to $\mathcal{D}^H_{sk_i}$, A_{cpa} runs the knowledge extractor K as follows.
  - In the find stage, if $j' = j$ then B_j returns c and halts; otherwise, A_{cpa} runs K as $m \leftarrow K(T_H, \epsilon, c, pk_i)$ and returns m to A_{cca}.
  - In the guess stage, if $j' = j$ then B_j returns c and halts; otherwise, A_{cpa} runs K as $m \leftarrow K(T_H, \hat{c}, c, pk_i)$ and returns m to A_{cca}.

Since $j \leq q_d$ and A_{cca} makes at most q_d queries to the decryption oracles, B_j is sure to output c and halt before A_{cca} outputs d in the guess stage.

We analyze the success probability of A_{cpa}. We have that for any $j \in \{1, 2, \cdots, q_d\}$ the distribution of $(T_H, C, c, pk_i) \leftarrow \mathtt{run}\ B_j^{H, \mathcal{E}^H_{pk_0}, \mathcal{E}^H_{pk_1}}(pk_0, pk_1, i)$ where

$$H \leftarrow \Omega;\ I \leftarrow \mathcal{G}(1^k);\ (pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathcal{K}(I)$$

and the distribution of the j-th input for K in the above adversary A_{cpa} is identical. Therefore,

$$\Pr[A_{\mathrm{cpa}}(\mathsf{find}, pk_0, pk_1) = A_{\mathrm{cca}}(\mathsf{find}, pk_0, pk_1)] \geq 1 - q_d^{\mathrm{find}} \cdot (1 - \lambda)$$

and

$$\begin{aligned}\Pr[&A_{\mathrm{cpa}}(\mathsf{guess}, c, (\mathsf{si}, T_H)) = A_{\mathrm{cca}}(\mathsf{guess}, c, \mathsf{si}) \\ &|A_{\mathrm{cpa}}(\mathsf{find}, pk_0, pk_1) = A_{\mathrm{cca}}(\mathsf{find}, pk_0, pk_1)] \geq 1 - (q_d - q_d^{\mathrm{find}}) \cdot (1 - \lambda)\end{aligned}$$

where q_d^{find} is a number of decryption queries of A_{cca} in the find stage. Hence, $\epsilon' \geq \epsilon - 2q_d(1 - \lambda)$.

It is easy to see that the running time of A_{cpa} is less than $t_{\mathrm{cca}} + q_d \cdot t_{\mathrm{KETK}}$.

5 Fujisaki–Okamoto Conversion

In this section, we review the conversion proposed by Fujisaki and Okamoto [3].

Let $\Pi^{\mathrm{pub}} = (\mathcal{G}^{\mathrm{pub}}, \mathcal{K}^{\mathrm{pub}}, \mathcal{E}^{\mathrm{pub}}, \mathcal{D}^{\mathrm{pub}})$ be a public-key encryption scheme and let $\Pi^{\mathrm{sym}} = (\mathcal{E}^{\mathrm{sym}}, \mathcal{D}^{\mathrm{sym}})$ be a symmetric-key encryption scheme. Let $G : \mathtt{MSPC}^{\mathrm{pub}} \to \mathtt{KSPC}^{\mathrm{sym}}$ and $H : \mathtt{MSPC}^{\mathrm{pub}} \times \mathtt{MSPC}^{\mathrm{sym}} \to \mathtt{COINS}^{\mathrm{pub}}$ be hash functions.

A public-key encryption scheme $\Pi^{\mathrm{hy}} = (\mathcal{G}^{\mathrm{hy}}, \mathcal{K}^{\mathrm{hy}}, \mathcal{E}^{\mathrm{hy}}, \mathcal{D}^{\mathrm{hy}})$ derived from the Fujisaki-Okamoto conversion is as follows:

- Common key generation and key generation: $\mathcal{G}^{\mathrm{hy}}$ and $\mathcal{K}^{\mathrm{hy}}$ are the same as $\mathcal{G}^{\mathrm{pub}}$ and $\mathcal{K}^{\mathrm{pub}}$, respectively.
- Encryption:
$$\mathcal{E}^{\mathrm{hy}}_{pk}(m;\sigma) = \mathcal{E}^{\mathrm{pub}}_{pk}(\sigma; H(\sigma, m)) \,||\, \mathcal{E}^{\mathrm{sym}}_{G(\sigma)}(m)$$
where $\mathtt{COINS}^{\mathrm{hy}} = \mathtt{MSPC}^{\mathrm{pub}}$ and $\mathtt{MSPC}^{\mathrm{hy}} = \mathtt{MSPC}^{\mathrm{sym}}$.
- Decryption:
$$\mathcal{D}^{\mathrm{hy}}_{sk}(c_1||c_2) = \begin{cases} \hat{m} & \mathtt{if}\ c_1 = \mathcal{E}^{\mathrm{pub}}_{pk}(\hat{\sigma}; H(\hat{\sigma}, \hat{m})) \\ \perp & \mathtt{otherwise} \end{cases}$$
where $\hat{\sigma} \leftarrow \mathcal{D}^{\mathrm{pub}}_{sk}(c_1)$ and $\hat{m} \leftarrow \mathcal{D}^{\mathrm{sym}}_{G(\hat{\sigma})}(c_2)$.

Fujisaki and Okamoto showed that the public-key encryption scheme Π^{hy} is secure in the sense of IND-CCA in the random oracle model when

- Π^{pub} is γ-uniform ($\gamma < 1$) and secure in the sense of OW, and
- Π^{sym} is secure in the sense of FG.

6 Generic Conversion for the Anonymity

In this section, we propose the generic conversion for the anonymity, that is, we can prove that the public-key encryption scheme derived from the Fujisaki-Okamoto conversion with the following assumptions is secure in the sense of IK-CCA in the random oracle model.

- Π^{pub} use the common message space $\mathtt{MSPC}^{\mathrm{pub}}(I)$ and the common randomness space $\mathtt{COINS}^{\mathrm{pub}}(I)$ as the message space $\mathtt{MSPC}^{\mathrm{pub}}(pk)$ and the randomness space $\mathtt{COINS}^{\mathrm{pub}}(pk)$, respectively, for any public key pk outputted by $K(I)$,
- Π^{pub} is secure in the sense of IK-CPA,
- Π^{pub} is γ-uniform ($\gamma < 1$) and secure in the sense of OW, and
- Π^{sym} is secure in the sense of FG.

Since these conditions are sufficient that Π^{hy} meets IND-CCA, we can get a public-key encryption scheme which is secure in the sense of IND-CCA and IK-CCA in the random oracle model when we assume the above four conditions.

IK-CPA Security. We can prove the following lemma with respect to the anonymity property. See the full version [13] of this paper.

Lemma 1. *Let Π^{pub} be a public-key encryption scheme where Π^{pub} uses the common message space $\mathtt{MSPC}^{\mathrm{pub}}(I)$ and the common randomness space $\mathtt{COINS}^{\mathrm{pub}}(I)$ as the message space $\mathtt{MSPC}^{\mathrm{pub}}(pk)$ and the randomness space $\mathtt{COINS}^{\mathrm{pub}}(pk)$, respectively, for any public key pk outputted by $K(I)$.*

Suppose that Π^{pub} is (t_1, ϵ_1)-secure in the sense of IK-CPA, and (t_2, ϵ_2)-secure in the sense of OW. Let ℓ_2 be the size of $\mathtt{MSPC}^{\mathrm{sym}}$. Then, Π^{hy} is (t, q_g, q_h, ϵ)-secure in the sense of IK-CPA in the random oracle model, where $t = \min\{t_1, t_2\} - poly(\ell_2)$ and $\epsilon = \epsilon_1 + 2(q_g + q_h) \cdot \epsilon_2$.

Knowledge Extractor for PATK. We can show the existence of the knowledge extractor for PATK of our scheme.

Though we mentioned that we could not use the knowledge extractor for PA directly as that for PATK, fortunately, we can use the knowledge extractor for PA as that for PATK in the case of the Fujisaki-Okamoto conversion.

We can prove the following lemma. See the full version [13] of this paper.

Lemma 2. *Suppose that Π^{pub} is γ-uniform and (t_2, ϵ_2)-secure in the sense of OW. Suppose that Π^{sym} is (t_3, ϵ_3)-secure in the sense of FG. Let ℓ_1 and ℓ_2 be the sizes of $\mathtt{MSPC}^{\mathrm{pub}}$ and $\mathtt{MSPC}^{\mathrm{sym}}$, respectively. Then, there exist a $(t, \lambda, q_g, q_h, q_e)$-knowledge extractor K for PATK of Π^{hy} such that $t = (q_g + q_h) \cdot poly(\ell_1 + \ell_2)$ and $\lambda = 1 - 2q_e \cdot \epsilon_2 - 2\epsilon_3 - \gamma - 2^{-\ell_2}$.*

From Theorem 1 and Lemmas 1 and 2, we have the following theorem.

Theorem 2. *Let Π^{pub} be a public-key encryption scheme where Π^{pub} uses the common message space $\mathtt{MSPC}^{\mathrm{pub}}(I)$ and the common randomness space $\mathtt{COINS}^{\mathrm{pub}}(I)$ as the message space $\mathtt{MSPC}^{\mathrm{pub}}(pk)$ and the randomness space $\mathtt{COINS}^{\mathrm{pub}}(pk)$ for any public key pk outputted by $K(I)$, respectively.*

Suppose that Π^{pub} is γ-uniform, (t_1, ϵ_1)-secure in the sense of IK-CPA, and (t_2, ϵ_2)-secure in the sense of OW. Suppose that Π^{sym} is (t_3, ϵ_3)-secure in the sense of FG. Let ℓ_1 and ℓ_2 be the sizes of $\mathtt{MSPC}^{\mathrm{pub}}$ and $\mathtt{MSPC}^{\mathrm{sym}}$, respectively. Then, Π^{hy} is $(t, q_g, q_h, q_d, \epsilon)$-secure in the sense of IK-CCA in the random oracle model where $t = \min\{t_1, t_2\} - (q_g + q_h) \cdot poly(\ell_1 + \ell_2)$. and $\epsilon = \epsilon_1 + 2(q_g + q_h)\epsilon_2 + 2q_d(2\epsilon_2 + 2\epsilon_3 + \gamma + 2^{-\ell_2})$.

7 Concluding Remarks

The previously proposed public-key encryption schemes in [10, 11, 12] which are based on RSA-OAEP and secure in the sense of IK-CCA seem to meet PAKE.

It might be interesting to consider the definition of the plaintext awareness in the two-key setting without random oracles and the schemes in the standard model which meet the plaintext awareness in the two-key setting.

References

1. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption – How to Encrypt with RSA. In De Santis, A., ed.: Advances in Cryptology – EUROCRYPT '94. Volume 950 of LNCS., Perugia, Italy, Springer-Verlag (1994) 92–111
2. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is Secure under the RSA Assumption. [14] 260–274
3. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In Wiener, M., ed.: Advances in Cryptology – CRYPTO '99. Volume 1666 of LNCS., Santa Barbara, California, USA, Springer-Verlag (1999) 537–554
4. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. [14] 213–229
5. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among Notions of Security for Public-Key Encryption Schemes. In Krawczyk, H., ed.: Advances in Cryptology – CRYPTO '98. Volume 1462 of LNCS., Santa Barbara, California, USA, Springer-Verlag (1998) 26–45
6. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In Lee, P.J., ed.: Advances in Cryptology – ASIACRYPT 2004. Volume 3329 of LNCS., Jeju Island, Korea, Springer-Verlag (2004) 48–62
7. Phan, D.H., Pointcheval, D.: Chosen-Ciphertext Security without Redundancy. In Laih, C.S., ed.: Advances in Cryptology – ASIACRYPT 2003. Volume 2894 of LNCS., Taipei, Taiwan, Springer-Verlag (2003) 1–18
8. Cui, Y., Kobara, K., Imai, H.: A Generic Conversion with Optimal Redundancy. In Menezes, A., ed.: Topics in Cryptology – CT-RSA 2005. Volume 3376 of LNCS., San Francisco, CA, USA, Springer-Verlag (2005) 104–117
9. Fujisaki, E.: Plaintext-Simulatability. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Special Section on Cryptography and Information Security **E89-A** (2006) 55–65
10. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-Privacy in Public-Key Encryption. In Boyd, C., ed.: Advances in Cryptology – ASIACRYPT 2001. Volume 2248 of LNCS., Gold Coast, Australia, Springer-Verlag (2001) 566–582 Full version of this paper, available via `http://www-cse.ucsd.edu/users/mihir/`.
11. Hayashi, R., Okamoto, T., Tanaka, K.: An RSA Family of Trap-door Permutations with a Common Domain and its Applications. In Bao, F., Deng, R.H., Zhou, J., eds.: Public Key Cryptography – PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography. Volume 2947 of LNCS., Singapore, Springer-Verlag (2004) 291–304
12. Hayashi, R., Tanaka, K.: The Sampling Twice Technique for the RSA-based Cryptosystems with Anonymity. In Vaudenay, S., ed.: Public Key Cryptography – PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography. Volume 3386 of LNCS., Les Diablerets, Switzerland, Springer-Verlag (2005) 216–233
13. Hayashi, R., Tanaka, K.: PA in the Two-Key Setting and a Generic Conversion for Encryption with Anonymity. Research Report C-224, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, `http://www.is.titech.ac.jp/research/research-report/` (2006)
14. Kilian, J., ed.: Advances in Cryptology – CRYPTO 2001. Volume 2139 of LNCS., Santa Barbara, California, USA, Springer-Verlag (2001)

Statistical Decoding Revisited

R. 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 statistical decoding attack on the McEliece cryptosystem from [4]. The statistical decoding algorithm is a probabilistic algorithm for correcting errors in random codes. It uses precomptuations to provide faster error correction than the classical general decoding algorithms. We analyze the success probability of the algorithm and show how to improve it. Further, we show that the algorithm may not be used to attack the McEliece cryptosystem, due to the large amount of precomputation needed.

Keywords: McEliece Cryptosystem, general decoding, coding theory, public key cryptography, code based cryptography.

1 Introduction

The security of cryptosystems based on error correcting codes is connected to the hardness of the general decoding problem. The first cryptosystem, which is based on that technique is the one presented by McEliece in 1978. McEliece's cryptosystem is very effective in en- and decryption, has a good information rate and we can even build a signature scheme from it. Furthermore, despite all effort, it remains unbroken for large public key sizes.

The statistical decoding attack on the McEliece PKC is a general decoding attack. It uses a precomputed alternative description of the public key, which has exponential space complexity. The author of [4] claims, that this alternative description can be computed in reasonable time. We show, that this is not possible employing the method proposed by Al Jabri. As a consequence, the attack fails even for the original parameter set of the McEliece cryptosystem, which is insecure against general decoding attacks [2].

However, statistical decoding can be used to correct errors in short random codes. After some precomputation, statistical decoding corrects errors more efficiently than the standard general decoding algorithms. Its mayor disadvantage is, that the algorithm is probabilistic and fails in some cases. We show how to improve the probability of correct decoding in that case.

The paper is structured as follows: In this section we give an introduction into the basic concepts of coding theory and the McEliece cryptosystem. In the

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 283–294, 2006.
© Springer-Verlag Berlin Heidelberg 2006

second and third section we present the statistical decoding algorithm and show, how to improve it. In the fourth sections we analyze the precomputation phase of the statistical decoding algorithm.

1.1 Coding Theory and Problems

The security of the McEliece cryptosystem is based on the difficulty of some classical problems of coding theory. Here we give a short introduction into the topic of coding theory.

Definition 1.1. *An (n,k)-code $\mathcal{C}$ over a finite field $\mathbb{F}$ is a k-dimensional subvectorspace of the vector space $\mathbb{F}^n$. We call $\mathcal{C}$ an (n,k,d)-code if the minimum distance is $d = \min_{\mathbf{x},\mathbf{y}\in\mathcal{C}} \mathrm{dist}\,(\mathbf{x},\mathbf{y})$, where "*dist*" denotes the Hamming distance. The distance of an element $\mathbf{x} \in \mathbb{F}^n$ to the null-vector $\mathrm{wt}\,(\mathbf{x}) := \mathrm{dist}\,(\mathbf{0},\mathbf{x})$ is called weight of $\mathbf{x}$.*

Definition 1.2. *The matrix $\mathsf{C} \in \mathbb{F}^{k\times n}$ is a generator matrix for the (n,k) code $\mathcal{C}$ over $\mathbb{F}$, if the rows of C span $\mathcal{C}$ over $\mathbb{F}$. The matrix $\mathsf{H} \in \mathbb{F}^{(n-k)\times n}$ is called check matrix for the code $\mathcal{C}$ if $\mathsf{H}^\top$ is the right kernel of C. The code generated by H is called dual code of $\mathcal{C}$ and denoted by $\mathcal{C}^\perp$.*

With these definitions, we are able to define the problems of coding theory on which the security of the McEliece cryptosystem rely. The $\mathcal{NP}$-hardness of these problems is proven e.g. in [1].

Definition 1.3. *The general decoding problem for linear codes is defined as follows: For a given (n,k) linear code $\mathcal{C}$ over $\mathbb{F}$ and a vector $\mathbf{y} \in \mathbb{F}^n$ find $\mathbf{x} \in \mathcal{C}$, where $\mathrm{dist}\,(\mathbf{y},\mathbf{x})$ is minimal.*

Let $\mathbf{e}$ be a vector of weight $\leq t := \left\lfloor \frac{d-1}{2} \right\rfloor$ and $\mathbf{x} \in \mathcal{C}$. Then there is a unique solution to the general decoding problem for $\mathbf{y} = \mathbf{x} + \mathbf{e}$. The code $\mathcal{C}$ is said to be an t-error correcting code.

Definition 1.4. *The problem of finding weights of a linear code is defined as follows: For a given linear code $\mathcal{C}$ over $\mathbb{F}$ and $w \in \mathbb{N}$ find a vector $\mathbf{x} \in \mathcal{C}$ with weight w.*

Throughout this paper, we will use the following notation. We write $\mathcal{G} = \langle \mathsf{G} \rangle$ if the linear (n,k)-code $\mathcal{G}$ over $\mathbb{F}$ has the generator matrix G. We can write $\mathbf{x} \in \mathcal{G}$ as $(x_1, \cdots, x_n) \in \mathbb{K}^n$. For any (ordered) subset $\{j_1, \cdots j_m\} = J \subseteq \{1, \cdots n\}$ we denote the vector $(x_{j_1}, \cdots, x_{j_m}) \in \mathbb{K}^m$ with $\mathbf{x}_J$. Similarly we denote by $\mathsf{M}_{.J}$ the submatrix of a $k \times n$ matrix M consisting of the columns corresponding to the indices of J and $\mathsf{M}_{J'.} = \left(\mathsf{M}^\top\right)_{.J'}$ for any (ordered) subset J' of $\{1, \cdots, k\}$.

1.2 The McEliece PKC

This cryptosystem was proposed by McEliece [5] and is the first, which uses error correcting codes as a trapdoor. It remains unbroken in its original version, which uses irreducible binary Goppa codes. There exist efficient algorithms to correct errors up to half of the designed minimum distance of the Goppa code. We briefly describe the cryptosystem:

- **System Parameters:** $n = 2^m$, $t \in \mathbb{N}$, where $t \ll n$.
- **Key Generation:** Given the parameters n, t generate the following matrices:
  G' : $k \times n$ generator matrix of a binary irreducible $(n, k = 2^m - mt, 2t + 1)$ Goppa code $\mathcal{G}$
  S : $k \times k$ random binary non-singular matrix
  P : $n \times n$ random permutation matrix
  Then, compute the $k \times n$ matrix $\mathsf{G} = \mathsf{S}\mathsf{G}'P$.
- **Public Key:** (G, t)
- **Private Key:** $(\mathsf{S}, D_{\mathcal{G}}, \mathsf{P})$, where $D_{\mathcal{G}}$ is an efficient decoding algorithm for $\mathcal{G}$.
- **Encryption:** To encrypt a plaintext $\mathbf{m} \in \{0,1\}^k$ choose a vector $\mathbf{z} \in \{0,1\}^n$ of weight t randomly and compute the ciphertext $\mathbf{c}$ as follows:

$$\mathbf{c} = \mathbf{m}\mathsf{G} \oplus \mathbf{z} \ .$$

- **Decryption:** To decrypt a ciphertext $\mathbf{c}$ calculate

$$\mathbf{c}\mathsf{P}^{-1} = (\mathbf{m}\mathsf{S})\,\mathsf{G}' \oplus \mathbf{z}\mathsf{P}^{-1}$$

  first, and apply the decoding algorithm $\mathcal{D}_{\mathsf{G}}$ for $\mathcal{G}$ to it. Since $\mathbf{c}\mathsf{P}^{-1}$ has a hamming distance of t to the Goppa code we obtain the codeword

$$\mathbf{m}\mathsf{S}\mathsf{G}' = \mathcal{D}_{\mathcal{G}}\left(\mathbf{c}\mathsf{P}^{-1}\right) \ .$$

  Let $J \subseteq \{1, \cdots, n\}$ be a set, such that $\mathsf{G}_{.J}$ is invertible, then we can compute the plaintext $\mathbf{m} = \left(\mathbf{m}\mathsf{S}\mathsf{G}'\right)_J \left(\mathsf{G}'_{.J}\right)^{-1} S^{-1}$

In its initial version from 1978, McEliece proposed to choose $m = 10$ and $t = 50$, i.e. using a $(1024, 524, 101)$ Goppa code. After the proposal of several general decoding attacks the parameters had to be modified. The fastest of these attacks was proposed in [2], compare section 4 and 5. Today parameter sets with $m = 11$ and $40 \leq t \leq 93$ are considered to be secure. There exists also a signature scheme based on the McEliece PKC (CFS, see [3]), which is as secure as the McEliece PKC with the same parameters. For CFS it is proposed to choose e.g. $m = 16$ and $t = 9$.

2 Statistical Decoding

This general decoding algorithm was presented by A Kh. Al Jabri in [4]. The idea of statistical decoding may be described as follows: Given an (n, k, d) code $\mathcal{G}$, we first compute a sufficiently large set $\mathcal{H}_w$ of dual vectors of weight w (i.e. an alternative description of $\mathcal{G} = \mathcal{H}_w^{\perp}$). In the following we assume that $w < n/2$. However all observations are analogous for $w > n/2$. Given a word $\mathbf{y} = \mathbf{x} + \mathbf{e}$, where $\mathbf{x} \in \mathcal{G}$ and $\mathrm{wt}(\mathbf{e})$ is small, we take a vector $\mathbf{h} \in \mathcal{H}_w$, where $\mathbf{y}\mathbf{h}^{\top} \neq 0$. As $\mathbf{x}\mathbf{h}^{\top} = 0$, the non-zero positions of $\mathbf{h}$ reveals some information about $\mathbf{e}$. (Let e.g. $\mathrm{wt}(\mathbf{e}) = 4$, then either one or three non-zero entries of $\mathbf{e}$ correspond to non-zero entries of $\mathbf{h}$). Collecting the information each of the different vectors $\mathbf{h} \in \mathcal{H}_w$ reveals, we are able to find $\mathbf{e}$ in some cases.

There are three major questions regarding this technique, which we will address in the following sections: "How to compute the set $\mathcal{H}_w$?" (section 4), "How to combine the information the vectors of $\mathcal{H}_w$ reveal about $\mathbf{e}$?" (this section) and "What is the probability of identifying $\mathbf{e}$?" (section 3). In section 3.2 we show how to improve the success probability of correct decoding.

Let $\mathcal{H}_w$ be a set of vectors of weight w of the dual space of the $(n, k, 2t+1)$ linear binary code $\mathcal{G}$ with generator Matrix G. Let $\mathbf{y}$ be the sum of a codeword $\mathbf{u}\mathsf{G} \in \mathcal{G}$ and a error vector $\mathbf{e}$ with weight at most t. A Kh. Al Jabri points out, that for randomly generated codes the probability that a value of 1 appears in the i-th position of $\mathbf{h} \in \mathcal{H}_w$ with $\mathbf{y}\mathbf{h}^T = 1$ depends on i being a erroneous position in the vector $\mathbf{y}$. We have an *odd error detection* in i if $\mathbf{y}\mathbf{h}^T = 1$ and $\mathbf{h}_i = 1$. Assume that we have an odd error detection in i, then let p_w^+ be the probability that i is a erroneous position and q_w^+ be the probability that i is a non-erroneous position. We can compute these probabilities as

$$p_w^+ = \frac{\sum_{m \text{ odd}}^{m \leq t} \binom{n-t}{w-m}\binom{t-1}{m-1}}{\sum_{m \text{ odd}}^{m \leq t} \binom{t}{m}\binom{n-t}{w-m}},\ q_w^+ = \frac{\sum_{m \text{ odd}}^{m \leq t} \binom{n-t-1}{w-m-1}\binom{t}{m}}{\sum_{m \text{ odd}}^{m \leq t} \binom{t}{m}\binom{n-t}{w-m}}.$$

Since $w < n/2$ the inequation $p_w^+ > q_w^+$ holds, although for large w the difference is small. We define $v_{\mathbf{y},w}^+ := \sum_{\mathbf{h} \in \mathcal{H}_w} \left(\mathbf{y}\mathbf{h}^T \mod 2\right)$. Then, for $i \in \{1, \cdots, n\}$ an (non-)error position the random variable

$$\frac{1}{v_{\mathbf{y},w}^+} \sum_{\mathbf{h} \in \mathcal{H}_w} \left(\mathbf{y}\mathbf{h}^T \mod 2\right) \mathbf{h}_i$$

is the relative frequency estimate for p_w^+ (q_w^+ respectively). Its variance is $\left(\sigma_w^+\right)^2 = p_w^+(p_w^+ - 1)/v_{\mathbf{y},w}^+$. Thus, we can recover $\mathbf{u}$ using algorithm 2.1 if $\mathcal{H}_w$ is chosen in a way so that we can distinguish between p_w^+ and q_w^+.

Algorithm 2.1. StatDec

Input: $\mathcal{H}_w$, $\mathbf{y}$.
Output: $\mathbf{u}$, the information vector.

$\mathbf{v} = \sum_{\mathbf{h} \in \mathcal{H}_w} \left(\mathbf{y}\mathbf{h}^\top \mod 2\right) \mathbf{h} \in \mathbb{Z}^n$.

choose $I = \{\text{positions of the } k \text{ smallest entries of } \mathbf{v}\}$ s.t. $\mathsf{G}_{.I}$ is invertible.

$\mathbf{u} = \mathbf{y}_I \mathsf{G}_{.I}^{-1}$

Al Jabri claims, that precomputing a set $\mathcal{H}_w$ with

$$|\mathcal{H}_w| = 625 \cdot 10^{-6} \cdot p_w^+ \left(1 - p_w^+\right) \epsilon^{-2} \tag{1}$$

vectors is sufficient for correct decoding [4]. The work factor for algorithm 2.1 is

$$\mathcal{O}\left(n \cdot |\mathcal{H}_w| + 2k^3 + kn\right)$$

binary operations having computed the set $\mathcal{H}_w$ in advance. The author of [4] claims that the latter can be done e.g. by the methods of [2]. However, computing the set $\mathcal{H}_w$ is solving problem 1.4, which is a $\mathcal{NP}$-hard problem in general. In addition, a set $\mathcal{H}_w$ of the desired size will not even exist if w is chosen too small. Goppa codes, as BCH codes and GRS codes have a weight distribution "close" to the expected weight distribution of random code, which is the binomial distribution [4]. Consequently, we get the following condition for $\mathcal{H}_w$:

$$|\mathcal{H}_w| \leq \binom{n}{w} 2^{-k}, \tag{2}$$

if we want to decode e.g. a random code or a Goppa code. We come back to this problem in section 4, but first we want to analyze the success probability of StatDec.

3 The Success Probability of Statistical Decoding

The first point of critique on the statistical decoding is its success probability. In our experiments for small parameter sets we had difficulties, to correct errors with a set $\mathcal{H}_w$ of size given in equation (1). It seems, that the set has to be about 2^{13} times larger than claimed by Al Jabri to allow correct decoding in most cases. We give a brief example: For a $(2^6, 40, 9)$ Goppa code (or a $(2^6, 40, 9)$ random code), Al Jabri's bound for $|\mathcal{H}_{17}|$ is $1 \leq |\mathcal{H}_{17}| \leq \binom{64}{17} 2^{-40} \approx 2^{10}$. However, one vector of the dual code can not be sufficient for correct decoding in most cases. Therefore we want to take a closer look at the success probability of statistical decoding. Later we show how to improve StatDec and give a small example.

In the following, we assume, that every set $\mathcal{H}_w$ consists of random vectors of weight w. If the vectors in $\mathcal{H}_w$ are somehow related, the probability for finding the correct error vector decreases.

3.1 The Initial Algorithm

We return to the notations of the previous section. On input $\mathcal{H}_w$ and $\mathbf{y}$ StatDec does only return the correct error vector if for some δ with $p_w^+ - 1 < \delta < p_w^+$ the following two conditions hold:

(i) For every error position i:

$$\mathbf{v}_i > (p_w^+ - \delta) v_{\mathbf{y},w}^+.$$

(ii) There are at least k non-error positions j, such that

$$\mathbf{v}_j < (p_w^+ - \delta) v_{\mathbf{y},w}^+.$$

We may assume, that $v^+_{\mathbf{y},w} \approx \frac{1}{2}|\mathcal{H}_w|$, and thus the probability, that a certain δ fulfills the first condition is smaller than

$$\mathcal{P} := \Phi\left(\delta/\sigma^+_w\right)^t = \Phi\left(\delta\sqrt{\frac{\frac{1}{2}|\mathcal{H}_w|}{p^+_w(p^+_w-1)}}\right)^t, \tag{3}$$

where Φ refers to the distribution function of the standardized normal distribution. Thus, we have to choose

$$2\left(\Phi^{-1}\left(\mathcal{P}^{1/t}\right)\right)^2 p^+_w(1-p^+_w)\delta^{-2} \leq |\mathcal{H}_w| \leq \binom{n}{w}2^{-k}. \tag{4}$$

Assume $k \approx (n-t)/2$, then it is probable, that half of the values $\mathbf{v}_j$ for non error positions j will be below their mean value $p^+_w v^+_{\mathbf{y},w}$. Thus, if there exists an δ for a given ciphertext y, such that the two conditions above are fulfilled, then it will probably be smaller than $|p^+_w - q^+_w|$. Since $\Phi^{-1}(0.95) = 1.65$ we conclude, that with a set of size

$$|\mathcal{H}_w| \approx 5.4 p^+_w(1-p^+_w)\frac{1}{(p^+_w - q^+_w)^2}. \tag{5}$$

we can correct errors with a probability about 0.95^t. Note, that this number is a factor 2^{13} larger than the one given by Al Jabri. We expect that with a set of size given in equation (1) we could correct errors with a probability about $1/2^t$, only.

3.2 An Improved Version

To improve the probability of correct error correction, we want to include *even error detection*. Let $\mathbf{y}$ be the sum of a codeword $\mathbf{u}\mathsf{G} \in \mathcal{G}$ and a error vector $\mathbf{e}$ with weight at most t. We observe, that for randomly generated codes the probability that a value of 1 appears in the i-th position of $\mathbf{h} \in \mathcal{H}_w$ with $\mathbf{y}\mathbf{h}^T = 0$ depends on i being a erroneous position in the vector $\mathbf{y}$. Thus, we have an even error detection if $\mathbf{y}\mathbf{h}^T = 0$ and $\mathbf{h}_i = 1$. Let p^-_w be the probability that i is a erroneous position and q^-_w be the probability that i is a non-erroneous position in the case of an even error detection. These probabilities can be computed as follows:

$$p^-_w = \frac{\sum_{2\leq m \text{ even}}^{m\leq t}\binom{n-t}{w-m}\binom{t-1}{m-1}}{\sum_{m \text{ even}}^{m\leq t}\binom{t}{m}\binom{n-t}{w-m}}, \quad q^-_w = \frac{\sum_{m \text{ even}}^{m\leq t}\binom{n-t-1}{w-m-1}\binom{t}{m}}{\sum_{m \text{ even}}^{m\leq t}\binom{t}{m}\binom{n-t}{w-m}}.$$

We define $v^-_{\mathbf{y},w} := \sum_{\mathbf{h}\in\mathcal{H}_w}\left(1 - \mathbf{y}\mathbf{h}^T \mod 2\right)$. Then, for an (non-)error position i the value

$$\frac{1}{v^-_{\mathbf{y},w}}\sum_{\mathbf{h}\in\mathcal{H}_w}\left(1 - \mathbf{y}\mathbf{h}^T \mod 2\right)\mathbf{h}_i$$

is the relative frequency estimate for p^-_w (q^-_w respectively). We observe, that if $p^+_w > q^+_w$, then $p^-_w < q^-_w$.

For all possible weights, the relative frequency estimates of p_w^+ and p_w^- are approximately normal distributed if $|\mathcal{H}_w|$ is large enough. Therefore we can use the standard transformation, s.t. all the relative frequency estimates are $\mathcal{N}(0,1)$ distributed. It follows, that one can sum the scaled relative frequency estimates obtained by several sets containing dual vectors of different weights. As a consequence, we consider $\mathcal{H}$ as the set of all dual vectors of weight w satisfying $b \leq w \leq B < n/2$, i.e. $\mathcal{H} = \bigcup_{w=b}^{B} \mathcal{H}_w$. All in all, we get the modified algorithm 3.1. With the notation of StatDec+: If i is an error position, then for all $\mathbf{v}$, $(\mathbf{v})_i$ has mean value 0. For an implementation one should omit the previous computation of σ_w^+ and σ_w^-. and compute these values while computing $\mathbf{v}_w$.

Algorithm 3.1. StatDec+

Input: $\mathcal{H} = \bigcup_{w=b}^{B} \mathcal{H}_w$, $\mathbf{y}$.
Output: $\mathbf{u}$, the information vector.

for $w = b$ to B **do**
$\left(\sigma_w^+\right)^2 = p_w^+ \cdot (1 - p_w^+) \cdot v_{\mathbf{y},w}^+$.
$\left(\sigma_w^-\right)^2 = p_w^- \cdot (1 - p_w^-) \cdot v_{\mathbf{y},w}^-$.

$\mathbf{1} = (1, 1, \cdots, 1) \in \{0,1\}^n$.
for $w = b$ to B **do**
$\mathbf{v}_w = \sum_{\mathbf{h} \in \mathcal{H}_w} \left(\mathbf{y}\mathbf{h}^\top \mod 2\right) (\mathbf{h} - p_w^+ \mathbf{1})/\sigma_w^+ \in \mathbb{R}^n$.
$\mathbf{v}_{w+B} = -\sum_{\mathbf{h} \in \mathcal{H}_w} \left(1 - \mathbf{y}\mathbf{h}^\top \mod 2\right) (\mathbf{h} - p_w^- \mathbf{1})/\sigma_w^- \in \mathbb{R}^n$.

for all binary combinations $\mathbf{v}$ of the different $\mathbf{v}_l$ **do**
choose $I = \{$positions of the k smalles entries of $\mathbf{v}\}$ s.t. $\mathsf{G}_{.I}$ is invertible.
$\mathbf{u} = \mathbf{y}_I \mathsf{G}_{.I}^{-1}$
if weight$(\mathbf{u}\mathsf{G} \oplus \mathbf{y}) \leq t$ **then**
return $\mathbf{u} = \mathbf{u}$

Let us assume, that the different relative frequency estimates are independent. We define $\mathbf{v} = \sum_{w=b}^{B} e_w \mathbf{v}_w + \sum_{w=b}^{B} e_{w+B} \mathbf{v}_{w+B}$, where each $e_i \in \{0,1\}$. Then for an error position j, $(\mathbf{v})_j$ is normal distributed with mean value 0 and variance σ^2 equal to the number of $e_w \neq 0$. If j is a non-error position, then $(\mathbf{v})_j$ is normal distributed with mean value

$$E := \sum_{w=b}^{B} e_w \left(\frac{|q_w^+ - p_w^+|}{\sigma_w^+} v_{\mathbf{y},w}^+ \right) + \sum_{w=b}^{B} e_{w+B} \left(\frac{|q_w^- - p_w^-|}{\sigma_w^-} v_{\mathbf{y},w}^- \right)$$

and variance

$$S^2 = \sum_{w=b}^{B} w_w \left(\frac{q_w^+(1-q_w^+)}{\left(\sigma_w^+\right)^2} v_{\mathbf{y},w}^+ \right) + \sum_{w=b}^{B} w_{w+B} \left(\frac{q_w^-(1-q_w^-)}{\left(\sigma_w^-\right)^2} v_{\mathbf{y},w}^- \right)$$

In most cases we will have $2v_{\mathbf{y},w}^+ \approx 2v_{\mathbf{y},w}^- \approx |\mathcal{H}_w|$. To distinguish between error and non-error positions by $\mathbf{v}$, we get the following conditions: There exists an

$\delta \in \mathbb{R}$ such, that for every error position i the inequation $|\mathbf{v}_i| < \delta$ holds and there are at least k non-error positions j, such that $|\mathbf{v}_j| > \delta$. The probability, that a certain δ fulfills this conditions is smaller than $\Phi\left(\delta/\sigma\right)^t$. Again, we expect, that the condition $\delta \leq E$ has to be true in most cases, and thus we get

$$\mathcal{P} \approx \Phi\left(\frac{1}{\sigma}\left(\sum_{w=b}^{B} e_w \sqrt{\frac{\left|q_w^+ - p_w^+\right|^2 |\mathcal{H}_w|}{2p_w^+(1-p_w^+)}} + \sum_{w=b}^{B} e_{w+B} \sqrt{\frac{\left|q_w^- - p_w^-\right|^2 |\mathcal{H}_w|}{2p_w^-(1-p_w^-)}}\right)\right)^t$$

as a suitable estimate for the probability of correct decoding with StatDec+. However we are not able to prove, that the different relative frequency estimates for p_w^+ and q_w^+ are independent. Nevertheless, for an implementation it seems recommendable, to start with the vectors $\mathbf{v}$ where $|\{e_i \neq 0\}|$ is large.

3.3 Experimental Results

We made several experiments with codes of small length. As expected, the proposed variant StatDec+ of the initial algorithm allows error correction in a significant larger number of cases than StatDec, especially when the size of the sets $\mathcal{H}_w$ is small. Further, it seems recommendable to include sets $\mathcal{H}_w$ with small w, even if their size is smaller than desired (e.g. up to a factor 4).

In the following we present three examples of our experiments. Note that for all our examples the bound for $|\mathcal{H}_w|$ given by equation (1) is useless, as it is smaller than 0. Further, the precomputation to find the sets $\mathcal{H}_w$ was quite time-consuming and an exhaustive search in some cases. The time needed to perform the precomputation for StatDec+ is the same as for StatDec.

In our first example we considered a $(2^6, 40, 9)$ Goppa code. For this code the relative frequency estimates and the desired sizes of each $\mathcal{H}_w$ are given by table 3.1. We computed a set $\mathcal{H} = \{\mathcal{H}_{16}, \mathcal{H}_{17}, \mathcal{H}_{18}\}$, where each of the sets $\mathcal{H}_w$

Table 3.1. Correcting errors of weight 4 in a $(64, 40)$ code

w	p_w^+	q_w^+	p_w^-	q_w^-	$\lvert\mathcal{H}_w\rvert$
16	0.295	0.248	0.210	0.263	1433
17	0.302	0.263	0.232	0.268	2160
18	0.311	0.280	0.254	0.284	3393

consisted of 100 random vectors. With StatDec+ we were able to correct errors of weight 4 in 93.2% of the cases. With the original algorithm, called with each set $\mathcal{H}_w$, correct error correction was possible in 17.5% of the cases, only.

In the second example, we looked at the same code as in the first example, but chose each $\mathcal{H}_w$ to be the set of all vectors of weight w. For our particular Goppa code, we got: $|\mathcal{H}_{16}| = 345$, $|\mathcal{H}_{17}| = 1234$ and $|\mathcal{H}_{18}| = 3149$. In this case, error correction was possible with StatDec and StatDec+ in all cases. An correct error correction with StatDec would not have been possible in all cases, if only one of the sets $\mathcal{H}_w$ would have been used.

Table 3.2. Correcting errors of weight 6 in a $(64, 22)$ code

w	p_w^+	q_w^+	p_w^-	q_w^-	$\lvert\mathcal{H}_w\rvert$	StatDec success rate
8	0.183	0.119	0.082	0.129	562	95.0%
9	0.189	0.136	0.102	0.145	835	79.4%
10	0.196	0.152	0.122	0.160	1283	73.8%

In our last example, we looked at a $(2^6, 22, 13)$ random code. The values for the relative frequency estimates and the sizes of $\mathcal{H}_w$ resulting from equation (5) are given by table 3.2. The expected success probability of StatDec is $\approx 0.95^6 = 73.5\%$ for each set $\mathcal{H}_w$. However, the experimented success probability for StatDec is larger, compare table 3.2. In this case we were able to compute the desired sets in reasonable time. Again, we made 1000 attempts to correct errors of weight 6. With StatDec+ we were able to correct all errors, whereas with StatDec we would have been able to correct them in 99.2% of the cases.

4 On the Problem of Finding Weights

Al Jabri proposes to use a variant of Sterns algorithm to solve the problem of finding weights, i.e. to compute $\mathcal{H}_w$. J. Stern designed his algorithm to find a (unique) shortest codeword of a binary linear code. Such an algorithm can be used to correct up to $t := (d-1)/2$ errors in a binary (n, k, d) code $\mathcal{G}$: Let c be a binary n-vector with distance t to $\mathcal{G}$ and G be the generator matrix of $\mathcal{G}$. Then the sum of c and the unique shortest codeword of the code generated by

$$\left(\frac{G}{c}\right)$$

is the solution to the general decoding problem for G and c.

We recall the original algorithm of Stern [7], which tries to find a vector of low weight w. Let H be the check matrix of the code G. Given the parameters p and l, successively choose two disjoint sets of $p < k/2$ columns $\mathcal{I}_1$ and $\mathcal{I}_2$ at random. Then choose a set $\mathcal{J} \subseteq \{1, \cdots, n\} \setminus (\mathcal{I}_1 \cup \mathcal{I}_2)$ of l rows at random. We may assume without loss of generality, that $\mathcal{I}_1 = \{n-k+1, \cdots, n-k/2\}$ and $\mathcal{I}_1 = \{n-k/2+1, \cdots, n\}$. If we can not transform the matrix H into a systematic matrix, the algorithm fails at this point, and is started anew. Otherwise we transform H into the desired form. Now we may assume, that $\mathcal{J} = \{1, \cdots, l\}$ and get a check matrix of the following form:

$$\mathsf{H} = \left(\mathsf{Id}_{n-k} \middle| \frac{\mathsf{Z}_1 | \mathsf{Z}_2}{\mathsf{B}}\right),$$

where Z_1 and Z_2 are $l \times k/2$ matrices, and B is a $(k-l) \times k$ matrix. For all pairs of vectors $(\mathbf{e}_1, \mathbf{e}_2) \in (\{0,1\}^{k/2})^2$ where $\mathrm{wt}(\mathbf{e}_1) = \mathrm{wt}(\mathbf{e}_2) = p$ we check whether $\mathbf{e}_1\mathsf{Z}_1 = \mathbf{e}_2\mathsf{Z}_2$. If the condition is fulfilled for such a pair, then we compute the unique vector $\mathbf{e}_0 \in \{0,1\}^{n-k}$, such that $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)\,\mathsf{H}^\top = 0$. The vector

$\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$ is our candidate for a short codeword. One can observe, that the fist l entries of $\mathbf{e}$ are zeros and thus the weight of $\mathbf{e}$ is smaller than $n-k-l+2p$. If none of the constructed vectors $\mathbf{e}$ is of the desired weight, then the algorithm fails. The success probability of one iteration of the algorithm is

$$\mathcal{P}_{p,l,w} = \frac{\binom{n-w}{k/2-p}\binom{w}{p}\binom{n-w-k/2-p}{k/2-p}\binom{w-p}{p}\binom{n-k-(w-2p)}{l}}{\binom{n}{k/2}\binom{n-k/2}{k/2}\binom{n-k}{l}}$$

in the case of a unique code word $\mathbf{e}'$ of weight w.

To improve the performance of Sterns algorithm, one can view its dual variant – depending on the ratio of k/n – and try to avoid the costly Gaussian elimination by choosing $\mathcal{I}_1$ and $\mathcal{I}_2$ iteratively and not at random. This method was introduced and analyzed by Canteaut and Chabaud, compare [2]. The success probability of the algorithm for finding the shortest codeword is to be modeled by a Markov chain in that case. We omit details and just take the result, that the work factor for one iteration becomes

$$\Omega_{p,l} = \left(\frac{1}{2}n(n-k) + 2l\binom{k/2}{p}(p-1) + (n-k-l)(2p-1)\binom{k/2}{p}^2\frac{1}{2^l}\right).$$

The work factor of the resulting algorithm may be approximated by

$$\mathcal{O}(n^3)2^{-t\log_2(1-k/n)},$$

if t is small and k/n is not too close to 1 (compare [6]).

In the case of statistical decoding we use the algorithm from [2] not to find a single lowest weight code word, but several code words of a certain weight w. If there are several code words of weight w, the work factor decreases by a factor equal to the number of such code words. As the expected number of vectors of weight w is given by the binomial distribution, we get the expected workfactor to compute a set $\mathcal{H}_w$ of vectors of weight w as

$$\mathcal{W}_{p,l,w} = \frac{2^k}{\binom{n}{w}}\frac{\Omega_{p,l}}{\mathcal{P}_{p,l,w}} \cdot \sum_{i=0}^{|\mathcal{H}_w|-1}\left(1 - \frac{i\cdot 2^k}{\binom{n}{w}}\right)^{-1}. \tag{6}$$

If one wants to compute a set $\mathcal{H}$, which serves as an input for the StatDec+, we expect, that every execution of a single round of the algorithm returns

$$\sum_{w=b}^{B}\frac{2^k}{\binom{n}{w}}\mathcal{P}_{p,l,w}^{-1}$$

vectors of weight w satisfying $b \le w \le B$. However, using the algorithm from [2] might not always be the best choice to use, when trying to find multiple words of any given weight, even if we did not find a better way to do so.

Unfortunately we were not able to find an example parameter set, where the precomputation required for StatDec could be performed in less time than the one needs for a single call of Canteaut's and Chabaud's general decoding algorithm.

5 Attacking the McEliece PKC with Statistical Decoding

To our knowledge, the best way to attack the McEliece PKC is the attack proposed by Canteaut and Chabaud [2], see section 4. Since for the McEliece cryptosystem $n = 2^m$ and $k = n - tm$, N. Sendrier concludes, that the maximum degree of security for the McEliece cryptosystem against the general decoding attack from [2] is obtained for an information rate $k/n \approx 1 - 1/\exp(1)$ [6]. This would lead e.g. to the choice of $m = 11$ and $t \approx 70$ for the McEliece cryptosystem.

To attack the McEliece PKC with parameters $m = 10$ and $t = 50$ with statistical decoding, Al Jabri claims that computing a set $\mathcal{H}_w$ consisting of 2^{38} vectors is sufficient. Unfortunately Al Jabri does not name w, but we are quite sure, that he referred to the set $\mathcal{H}_{133}$. However, equation (3) implies, that the probability of correct decoding is about 2^{-50} in that case. A decoding attempt with StatDec takes 2^{48} binary operations for this input. Consequently, one would expect, that it would take approximately 2^{98} binary operations, before an attack on one of 2^{50} given ciphertexts is successful.

We have shown, that an attacker would need a set $\mathcal{H}_{137}$ consisting of approximately 2^{51} vectors to attack ciphertext of the McEliece PKC with parameters $n = 10$ and $t = 50$. Even storing a set of this size seems impossible nowadays and the work factor for a single decoding attempt would be larger than 2^{61}, which is not much faster than the general decoding algorithm of Canteaut and Chabaud [2]. However, it takes at least 2^{152} binary operations to compute the set $\mathcal{H}_{137}$ with the algorithm proposed by Canteaut and Chabaud. For this parameter set, one iteration for $l = 19$ and $p = 2$ of the algorithm requires about 2^{24} binary operations. Most of the vectors returned by the algorithm will be of weight 241. For each one of 2^{-17} iterations, we will get only one of those vectors. Thus, after performing 2^{80} Operations, one will still have computed less than 2^{39} vectors of weight 241. Having a range of $114 \leq w \leq 241$, we will have still have not enough vectors of the dual space to attack the McEliece cryptosystem. Thus, it is not possible to attack the McEliece cryptosystem with StatDec or StatDec+.

Table 5.3. StatDec for example parameter sets

McEliece parameters $(2^m, k, d = 2t + 1)$	w	$\lvert p_w^+ - q_w^+ \rvert$	$\lvert \mathcal{H}_w \rvert$	$\binom{n}{w} 2^{-k}$	Workfactor StatDec	Workfactor precomput.
$(1024, 524, 101)$	137	$0.2 \cdot 10^{-7}$	2^{51}	$2^{52.5}$	2^{61}	2^{152}
$(1024, 524, 101)$	153	$0.21 \cdot 10^{-8}$	2^{58}	2^{94}	2^{68}	2^{138}
$(2048, 1278, 141)$	363	$0.41 \cdot 10^{-14}$	2^{96}	$2^{96.9}$	2^{107}	2^{609}
$(65536, 65392, 9)$	32000	$0.17 \cdot 10^{-13}$	2^{93}	$2^{109.7}$	2^{109}	$>> 2^{131}$

The situation for the signature scheme CFS is the same: Any set, that would allow correct decoding in a non-negligible fraction of the cases is to big to be stored efficiently and it is infeasible to perform the precomputation (compare Table 5.3).

6 Conclusion

We have shown, how to improve the probability of correct error correction of the statistical decoding algorithm. We have performed experiments and have shown, that statistical decoding can be used for fast decoding of random linear codes after some precomputation. Nevertheless, we needed several sets of vectors, each about 2^{13} times larger than claimed by Al Jabri. Additionally the problem how to perform the precomputation efficiently remains unsolved. We conclude, that it is not possible to attack the McEliece cryptosystem (or the CFS signature scheme) with reasonable parameter sets by statistical decoding. However, there might exist non-standard parameter sets for the McEliece cryptosystem, which can be attacked by statistical decoding.

References

1. E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.
2. A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEETIT: IEEE Transactions on Information Theory*, 44, 1998.
3. N. Courtois, M. Finiasz, and N.Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248, pages 157–174. Springer-Verlag, 2001.
4. A. Al Jabri. A statistical decoding algorithm for general linear block codes. In *Cryptography and Coding 2001*, volume 2260 of *LNCS*, pages 1–8. Springer Verlag, 2001.
5. R.J. McEliece. A public key cryptosystem based on algebraic coding theory. *DSN progress report*, 42-44:114–116, 1978.
6. N. Sendrier. On the security of the McEliece public-key cryptosystem. In M. Blaum, P.G. Farrell, and H. van Tilborg, editors, *Proceedings of Workshop honoring Prof. Bob McEliece on his 60th birthday*, pages 141–163. Kluwer, 2002.
7. J. Stern. A method for finding codewords of small weight. *Coding Theory and Applications*, 388:106–133, 1989.

Towards Provable Security for Ubiquitous Applications

Mike Burmester*, Tri Van Le*, and Breno de Medeiros

Department of Computer Science, Florida State University
Tallahassee, Florida 32306-4530
{burmester, levan, breno}@cs.fsu.edu

Abstract. The emergence of computing environments where smart devices are embedded pervasively in the physical world has made possible many interesting applications and has triggered several new research areas. Mobile ad hoc networks (MANET), sensor networks and radio frequency identification (RFID) systems are all examples of such pervasive systems. Operating on an open medium and lacking a fixed infrastructure, these systems suffer from critical security vulnerabilities for which few satisfactory current solutions exist, particularly with respect to availability and denial-of-service. In addition, most of the extant knowledge in network security and cryptography cannot be readily transferred to the newer settings which involve weaker devices and less structured networks.

In this paper we investigate the security of pervasive systems and focus on availability issues in malicious environments. We articulate a formal security framework that is tuned for the analysis of protocols for constrained systems and show how this can be used with applications that involve MANET and RFID systems. In our approach we shall use optimistic protocols for which the overhead is minimal when the adversary is passive. When the adversary is active, depending on the application, the additional cost is either used to trace malicious behavior or born by non-constrained components of the system. Our goal is to design mechanisms that will support self-healing and promote a fault-free system state, or a stable system state, in the presence of a Byzantine adversary.

Keywords: Ubiquitous applications, RFID, MANET, fault tolerance, tracing malicious faults.

1 Introduction and Motivation

We investigate the security of pervasive systems with focus on availability issues in the presence of Byzantine faults. Our goal is to specify formal simulation frameworks for analyzing security objectives and more importantly to design novel mechanisms and algorithms that achieve proven availability, uninterrupted services, high efficiency and low overhead in such systems. With a large amount of research already invested into other non-security issues, it is preferable whenever

* The author was partly supported by NSF grants DUE 0243117 and CNS 0209092.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 295–312, 2006.
© Springer-Verlag Berlin Heidelberg 2006

possible to design mechanisms that integrate security into existing algorithms that are well established in the literature.

The emergence of computing environments where smart devices are embedded pervasively in the physical world has made possible many interesting applications and triggered several new research areas. These include pervasive systems with constrained resources that intelligently configure and connect themselves, in particular, mobile ad hoc networks (MANETs), sensor networks and RFID systems. Nevertheless, the deployment of such systems in practice poses great challenges concerning their security and robustness in the presence of malicious faults. So far, research has focused on functionality, performance and services, with security being given a lower priority and centered mainly on confidentiality and integrity, but not availability (under malicious attacks). In particular, most research on network security and cryptography involves highly powered, highly structured redundant systems. For this reason the proposed security solutions are often inappropriate for these networks. This is particularly true regarding Denial-of-Service (DoS) attacks against power constrained systems. Furthermore, most practical security patches are only for specific attacks, leaving an unjustified belief that proven security and efficiency are conflicting goals.

Overview of our results. We investigate secure mechanisms and protocols for ubiquitous systems. We consider two applications.

1. Secure Mobile Ad hoc Networks (MANET). In particular we:
   - Develop a formal simulation framework for security that focuses on availability under DoS attacks, and that allows for concurrency and universal composability.
   - Investigate mechanisms that provably support availability in malicious environments within the formal simulation security framework.
   - Design protocols that provide message-delivery guarantees and that provably support network self-healing from malicious attacks. In particular, protocols that enable migration of the network to a fault-free state.
2. Secure Radio Frequency Identification (RFID). In particular we:
   - Develop a formal simulation framework for security of RFID systems that models availability, privacy, and authenticity services, and
   - Design provably secure scalable anonymous authentication protocols for RFIDs in the formal framework.

Our approach is holistic and promotes self-healing. It provides for novel optimistic mechanisms that deal with security and availability issues in an efficient manner with low overhead. We focus on systems that will tolerate, trace and eliminate faults by reconfiguring. The threat model allows for a very powerful Byzantine adversary: malicious nodes are not bounded by the system specifications and can use covert channels, more powerful transmitters/receivers, responders etc. Security will be proven using a well established cryptographic framework. In contrast to traditional cryptographic solutions, our approach is suitable for low power, low cost devices in a malicious environments with no infrastructure.

A self-healing strategy for ubiquitous systems. Self-healing is achieved by tracing malicious behavior. Whenever a component exhibits non-system faulty behavior, a tracing mechanism is activated and the component isolated. Assuming the number of potential non-system faults is bounded (the Byzantine assumption), the system ultimately will be fault-free. By using low level (optimistic) mechanisms, there is no extra cost when the adversary is passive. Even when faults do occur, the overhead involved is small. This makes it possible to achieve our security goal with practical systems that have constrained resources.

2 Securing MANET Applications

2.1 A Formal Simulation Framework for Security

There are several ways to capture the unpredictable nature of a mobile ad hoc network. Whichever way is used, there are important mobility and medium aspects that must be reflected. In its simplest form, a *mobile ad hoc network* is a stochastic process $\mathcal{G} = \mathcal{G}_1, \mathcal{G}_2, \ldots$, where $\mathcal{G}_t$ is a random graph with node set V, for which communication is: (*i*) *synchronous*, the time for a single transmission to be received is bounded by a constant; (*ii*) *promiscuous*, a packet transmitted by node will be received by all its neighbors. Links can be undirected (the neighbor relationship is symmetric) or directed (the neighbor relationship is asymmetric). We note that to the best of our knowledge, to date, all routing algorithms proposed in the literature support only bidirectional or undirected links.

The stochastic aspect of $\mathcal{G}$ is determined by the states of the nodes of $\mathcal{G}$ and Nature. Nature's contribution comes from the environment and the fact that the communication is wireless. A wide variety of factors may affect the communication, ranging from weather to radio interference and physical obstacles.

Adversary structure. In our model for the adversary, malicious nodes are not bounded by the constraints of the mobile ad hoc network. In particular, the adversary can send packets to arbitrary nodes and eavesdrop on all communication (not necessarily via nodes under his control). In this respect, our model differs from other models.

Definition 1. Let Γ be a family of subsets V' of the node set V. We call Γ an *Adversary Structure* [20]. The adversary $\mathcal{A} = \mathcal{A}_\Gamma$ selects a subset $V' \in \Gamma$ and can corrupt all its nodes during the lifetime of the system.[1] $\mathcal{A}dv$ controls the nodes of V' and may use them to undermine the security of the network. We call these nodes *corrupted* or *faulty* and refer to $\mathcal{A}dv$ as a Γ-adversary. The

[1] There are several generalizations of this model. One such generalization allows Γ to be dynamic: at regular intervals $\mathcal{A}dv$ can replace V' by $V'' \in \Gamma$, that is, release the nodes of $V' \backslash V''$ and replace them by the nodes of $V'' \backslash V'$. Another generalization involves hybrid faults: malicious faults and physical faults. We shall not consider these models here.

adversary may be *passive* or *active*. A *passive* adversary (also called *honest-but-curious*) will only eavesdrop on the network communication. An *active* adversary may use the corrupted nodes to prevent the normal functioning of the network via *snooping, dropping, modifying*, and/or *fabricating* network messages. Nodes that are actively involved in such attacks and the corresponding faults are called *malicious* or *Byzantine*. Malicious nodes may use hidden (covert) channels or "wormholes" through which they can communicate or tunnel packets.

A particular case of the Adversary Structure model is the *Byzantine faults* model [36] for which $\Gamma = \{V' \subset V \mid |V'| \leq k\}$, for some threshold k. In this case the adversary $\mathcal{A} = \mathcal{A}_k$ can control up to k nodes. We call $\mathcal{A}_k$ a k-adversary.

Simulation framework. Our security simulation framework follows cryptographic paradigm for the security of network protocols [3]. Mobile nodes are probabilistic Turing machines with a special transceiver tape linked to a network oracle $\mathcal{O}_{\mathcal{G}}$. For concurrent executions of distributed algorithms, we use a model with an infinite collection of oracles that emulate concurrent sessions of the algorithm, with which the adversary can interact. The following definition captures the basic security requirements of this approach for secure distributed applications. For a more formal discussion of a simulation framework for security, see Section 5.

Definition 2. Let $\mathcal{G}$ be a mobile ad hoc network and π a distributed algorithm of $\mathcal{G}$.

- Γ-*availability* holds for π if $Prob[\pi$ fails in presence of $\mathcal{A}]$ is negligible for all Γ adversaries $\mathcal{A}$.
- Γ-*tolerance* holds for π if $|Prob[\pi$ fails in presence of $\mathcal{A}] - Prob[\pi$ fails in the absence of$\mathcal{A}]|$ is negligible for all Γ adversaries $\mathcal{A}$.

The probabilities are taken over the adversary's and honest parties' coin tosses.

2.2 Security Issues for Routing Algorithms

Communication in an ad hoc network is achieved by forwarding packets via paths. Depending on where most of the routing effort takes place, there are two types of routing: *network-centric* and *source-centric*. With network-centric routing (such as DSDV [31], WR [6] and AODV [32]) the routing effort is distributed within the network; with source-centric routing (such as DSR [26]) most of the routing effort is done by the source node.

From a security point of view, network-centric routing requires substantial cooperation between network nodes and strong trust relationships. These algorithms are therefore more vulnerable to malicious faults. On the other hand, with source-centric routing, the source is less dependent on intermediate node cooperation and thus less vulnerable to malicious attacks.

Denial-of-Service attacks. A DoS can be triggered in several ways, as for example by: (1) Flooding in dense networks; (2) Flooding irrelevant packets; (3) Packet dropping, *e.g.*, on routers; (4) Preventing route discovery.

Man-in-the-Middle attacks. In a MiM attack the adversary takes control of the communication channel between the source and destination by interposing between them. Active MiM attacks include, for instance: (1) Wormhole attacks [25]; (2) Rushing attacks [25]; and (3) Sybil attacks [17].

Security at the physical and data link layers. There are two types of faults that may occur in a routing algorithm: faults whose effect is stochastically indistinguishable from ordinary link failures caused by the mobility of the system, radio interference, power failure etc, and faults whose effect can be distinguished. Faults that do not deviate from ordinary failures are best dealt with at the physical or data link layer of the protocol stack with Medium Access Control protocols. At these layers one can also deal with jamming attacks (using frequency-hopping spread spectrum techniques) and most isolated DoS attacks.

Faults of the second type, although by definition statistically detectable, can be quite hard to trace or locate. They include malicious faults, which may occur when they are least expected, and may not be traceable with statistical failure analysis. The reason for this is that any analysis based on reported failures can be manipulated by the adversary. Faults of this type have to be addressed at higher levels.

Security issues of Ariadne, SEAD and SAODV. Several routing protocols in the literature address security issues (see *e.g.*, [30]). Examples that are well established in the literature include: Ariadne [23], SEAD [24] and Secure AODV [38]. These algorithms do not tolerate insider faults caused by packet dropping or by colluding nodes (on paths). In particular they do not tolerate wormhole attacks.

Tolerating DoS Attacks with cell grids and colored graphs. A cell-grid approach –see Figure 1,is proposed in [8] in which only one node in each cell is only needed to propagate a packet. This approach guarantees packet delivery during DoS attacks including malicious DoS attacks. This result is based on circular broadcast ranges. It is possible to remove this restriction and extend the cell-grid approach to routing protocols (for example, by taking a succession of adjacent cells as a virtual path).

2.3 An Optimistic Algorithm That Traces Malicious Faults

Here we describe an optimistic algorithm that will trace malicious faults [9]. For this algorithm there is no additional cost when there are no faults. When faults do occur, the cost to locate a fault is one tracing round and two digital signatures (for a short *probe* and a short $failreport$). In either case, a packet is confirmed successfully delivered, or a fault location is determined with only two digital signatures. This is the most efficient routing algorithm that will trace malicious behavior even when faulty nodes collude. It improves on the tracing algorithm in [2] that requires at least $\log(n)$ communication rounds and signatures to locate a malicious fault, and that does not consider collusions. In our algorithm faults that can be dealt with at the data link layer by error correction and re-sending

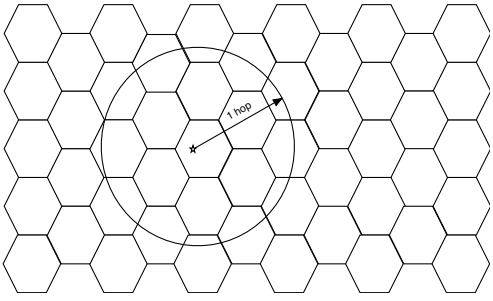


Fig. 1. The cell-grid and a node with its broadcast range

packets are treated as non-malicious. The protocol is described in Figure 2 and Figure 3. The following notation is used:

- $pkt_{sd} = [[s, d, sn, seq_s, data]]_{sd}$: a packet consisting of identifiers s, d, a session number sn for tracing algorithm (unique to each session), the sequence number seq_s for pkt_s, and $data$, authenticated with the key shared by s, d.
- $ack_{sd} = [[s, d, sn, seq_s]]_{sd}$: an authenticated acknowledgment.
- $prob_s = [s, d, sn, seq_s, hash(pkt_{sd}), hash(ack_{sd})]_s$: a digitally signed probing request by s.
- $failreport_y = [s, d, y, succ(y), sn, seq_s]_y$: a digitally signed failure report by y.
- $timer_{xy}$: a bound on time taken for a round trip from x to y for pkt_s.
- $prec(x)$, $succ(x)$: the node that precedes, succeeds x on the path taken by pkt_s.

In the protocol, the source s sends a packet pkt_{sd} to $succ(s)$ to be delivered to the destination d. If there are no faults then the packet reaches d and s will receive an authenticated acknowledgment ack_{sd}. If there is a fault the source s will send $prob_s$ with details of pkt_{sd} and ack_{sd} requesting from intermediate nodes to compare these values with their stored values. If a fault is detected by an intermediate node x then a $failreport_x$ is issued and send upstream to s. Each intermediate node x node sets timers $timer_{xd}$ and $timer_{xs}$, to determine if and when a $failreport_x$ should be issued. Thus if $succ(x)$ is not faulty, x will receive from $succ(x)$ after $x \xrightarrow{pkt_s} succ(x)$ and $x \xrightarrow{prob_s} succ(x)$ and before $timer_{xd}$ timeouts a valid $failreport_y$.

Observe that when there are no faults s and d only check the validity of pkt_{sd} and ack_{sd}, and intermediate nodes only forward pkt_{sd} and check the header of ack_{sd}.

Theorem 1 ([9]). *For any Γ-adversary, the tracing algorithm in Figures 2 and 3 either delivers pkt_s to the destination d or will trace at least one faulty node.*

Tracing malicious behavior with AODV and DSR. Most of the routing algorithms can easily be extended to incorporate our tracing mechanism in the communication phase. For example, for distance vector based routings such as

Source s. Set $seq_s = 0$. While a connection to d has not terminated do:

1. Set $timer_{sd}$ and send pkt_{sd} to $succ(s)$.
2. If a valid ack_{sd} for pkt_{sd} is received before timeout then set $seq_s = seq_s + 1$.
3. Else set $timer_{sd}$ and send $prob_s$ to $succ(s)$.
   (a) If a valid $failreport_y$ for pkt_{sd} is received before timeout then y or $succ(y)$ is malicious;
   (b) Else $succ(s)$ is malicious.

Destination d. When a valid pkt_s is received:

1. Construct and send ack_d to $prec(d)$.

Fig. 2. An optimistic tracing algorithm, I

Intermediate node x. When pkt'_{sd} is received:

1. Set $timer_{xs}$, and send pkt'_{sd} to $succ(x)$.
2. If an ack'_{sd} is received then send it to $prec(s)$.
   (a) If a valid $probe_s$ for pkt'_{sd} is received with $ack_{sd} \neq ack'_{sd}$ before $timer_{xs}$ times out, set $timer_{xd}$ and send $probe_s$ to $succ(x)$.
      i. If a valid $failreport_y$ for pkt'_{sd} is received before $timer_{xd}$ times out: send $failreport_y$ to $prec(x)$;
      ii. Else construct and send $failreport_x$ to $prec(x)$.
3. Else if a valid $probe_s$ for pkt'_{sd} is received with $pkt_{sd} = pkt'_{sd}$ before $timer_{xs}$ times out, reset $timer_{xd}$ and send $probe_s$ to $succ(x)$.
   (a) If a valid $failreport_y$ for pkt'_{sd} is received before $timer_{xd}$ timeouts: send $failreport_y$ to $prec(x)$;
   (b) Else construct and send $failreport_x$ to $prec(x)$.

Fig. 3. An optimistic tracing algorithm, II

DSDV, AODV, and DSR , malicious faults will be traced by using the optimistic tracing algorithm for packet processing (the store-and-forward process). This can be done at the network layer, *i.e.*, after error checking at the data link layer (MAC).

2.4 Adaptive Multipath Routing

In this section we propose to investigate secure distributed algorithms for finding maximal vertex disjoint paths that allow us to design secure AODV-type algorithms [32].

Multipath routing involves the establishment of multiple paths between source and destination pairs. These paths may be used for redundant communication to control malicious attacks. A major advantage in using multipaths is that, by exploiting redundancy we can guarantee service continuity, even when the adversary is active.

An Adaptive Multipath Routing algorithm. Finding routes with multiple paths in networks that do not have a fixed infrastructure is a challenge and in general requires a different approach to that used with fixed infrastructures. An

adaptive multipath routing algorithm that combines in parallel a distributed version of Ford-Fulkerson Max-Flow algorithm [18] (at the source) with a local network discovery algorithm (for nearby nodes) to find vertex-disjoint paths that link the source to the destination is proposed in [7]. This algorithm is proven secure in [9]. It is shown that:

Theorem 2. *The adaptive multipath routing algorithm in* [9] *tolerates any* k*-adversary, provided that the network graph is* $(k+1)$*-connected,* $k \geq 1$.

The novelty of this algorithm is that it is resistant to malicious DoS attacks which are addressed adaptively. In particular, when there are no attacks a single route is used. With each malicious attack, the multipath is adaptively reconstructed to deal with the threat. Only the shortest route(s) is (are) actually used, while the rest are kept alive. Furthermore, communication is activated as soon as a path becomes available, so there are no unnecessary delays.

In general when faults in a t-multipath occur beyond a certain acceptable threshold, the source s will use a $(t+1)$-multipath. Since the new set of paths is already constructed in the background, the delay caused by faults is minimized. Most of the time, there should be no delay. Furthermore, in our algorithm, the set of vertex-disjoint paths of the multipath is constructed incrementally, so that even when delays are unavoidable, they are minimal.

Observe that having local information available centrally is easier than having it distributed. In particular, the procedure used in the adaptive routing algorithm by the source allows more vertex-disjoint paths to be found than in most other multipath routing protocols. As a consequence fewer communication rounds may be needed when faults occur.

This algorithm can be can combined with the Dynamic Source Routing algorithm [26] to get an adaptive multipath DSR algorithm for reliability and service continuity in the presence of malicious adversary. Similarly, we may combine the adaptive multipath routing algorithm with the tracing mechanism to get an adaptive routing algorithm that will trace malicious behavior.

3 Securing RFID Applications

Radio Frequency Identification Devices (RFIDs) were initially developed as very small electronic hardware components having as their main function to broadcast a unique identifying number upon request. The simplest types of RFIDs are *passive tags*, that do not contain a power source, and are incapable of autonomous activity. These devices are powered by the reader's radiowaves, and the antenna doubles as a source of inductive power. Active tags, on the other hand, contain a power source and transmitter, and are capable of autonomous communication. Examples of such tags are toll passes. The low cost and high convenience value of RFIDs give them a potential for massive deployment, and it is expected that they will soon outnumber all other computing device types. Consequently, RFIDs are increasingly used in applications that interface with information security functions.

RFIDs are a challenging platform from an information assurance standpoint. Their extremely limited computational capabilities imply that traditional techniques for securing communication protocols cannot be used with such devices, and instead that new, lightweight approaches must be considered. Yet the privacy and security requirements of RFID applications can be quite significant. Herein we describe measures for the provision of security and privacy that are feasible for RFID applications. Ultimately, this should be accomplished with as rigorous a view of security as other types of applications.

It is our goal is to design protocols for RFID applications that:

1. are provably secure under formal simulation frameworks that capture the behavior of honest and adversarial parties, and that articulate a comprehensive security view in terms of an ideal functionality.
2. explicitly consider the existence of side-channels, an issue often ignored in modelling security of traditional applications, but which can be critically important in the RFID context; and
3. are computationally lightweight, taking into consideration the hardware-imposed constraints of the medium.

3.1 History of a Provably Secure RFID Authentication Protocol

In order to illustrate the need for comprehensive security models for RFID applications, we consider variants of the HB protocol which have been proposed as a practical form of secure RFID authentication. Introduced in [22], the HB protocol was originally designed for use by humans—who, as RFID tags, have limited ability to perform complex computations. In RFID tags, the HB protocol leverages the fact that generation of reasonably strong random numbers can be done cheaply by exploring physical properties—ultimately, exploiting the principle of little separation between hardware and software in the RFID domain.

The HB protocol can be proven secure against passive adversaries in a non-concurrent protocol execution setting by a simple reduction to the so-called "Learning Parity with Noise" (LPN) problem. However, it is completely insecure against an active adversary. To fix these problems, the HB protocol was adapted to include challenges from both readers and tags, leading to the HB+ protocol. Protocol HB+ can be proven secure against active adversaries [27] in a simplified model where the adversary is a malicious reader attacking a single honest tag. The proof has been generalized to a parallel and concurrent setting in [28] showing that rewinding techniques in the original security proof in [27] are not truly necessary; once re-winding is eliminated, one may claim as in [28] that multiple simulations can be executed simultaneously without exponential amplification of the probability of simulation failure, and therefore that the scheme is secure in the concurrent setting.

Both of the above security results are established in a simple attack model, which is not a multi-party model. They cannot be held as providing evidence that the scheme is secure in practical applications, where the adversary may communicate with both readers and tags simultaneously. Indeed, man-in-the-middle attacks do exist [19] and result in a total protocol break, as we now

describe. In what follows, we denote bit vectors in boldface, and if $\mathbf{v}$ is a bit vector, by $|\mathbf{v}|$ we mean the Hamming weight (number of non-zero components) of $\mathbf{v}$. If $\mathbf{a}$ and $\mathbf{b}$ are two bit vectors of length k, denote by $\mathbf{a} \cdot \mathbf{b}$ the value $a_1 \wedge b_1 \oplus \cdots \oplus a_k \wedge b_k$, where $\oplus$ denotes the XOR bit operation. Legitimate readers and tags share a pair of keys $(\mathbf{x}, \mathbf{y})$.

The attack works as follows: The attacker first chooses a k-bit string $\mathbf{d}$ at random and XORs it against the Reader's challenge $\mathbf{a}$ before forwarding it to the tag. Since the same value $\mathbf{d}$ is used in all protocol rounds until an outcome is produced by the Reader, it is easy to show that the authentication will be successful with overwhelming probability when $\mathbf{d} \cdot \mathbf{x} = 0$—for in that case the responses by the Tag are the same as in the regular protocol. The opposite will be true when $\mathbf{d} \cdot \mathbf{x} = 1$, i.e., the Reader will reject with overwhelming probability. The adversary has then learned one linear relation among the bits of $\mathbf{x}$, and by repeating the attack with different values of $\mathbf{d}$, the full value of $\mathbf{x}$ can be recovered. The adversary can also learn the key value $\mathbf{y}$ by performing a similar attack, where the value $\mathbf{d}$ is xored into the blinding factor $\mathbf{b}$ instead of into the challenge $\mathbf{a}$.

The attack illustrates the fact that security proofs, when carried out in an overly simplified attack model, fail to convey implications for the practical security of protocols. The above example-cum-moral-lesson-tale is a compelling one from the realm of recent RFID protocols, but the basic premise that protocol analysis should ideally be carried out within a comprehensive attack model has been well recognized—e.g., see [11] for general arguments in this regard.

There exist competing approaches for analyzing the security of protocols against arbitrary adversarial configurations. Some of these have been successful at other areas of computer science and adapted for the purposes of security analysis, such as techniques based on model-checking, and formal methods. We will take the approach of indistinguishability between real and ideal protocol simulations. This formalism is based on the premise that one should first define an *ideal functionality* for the protocol—i.e., how to achieve security in an ideal world where the honest parties have a secure communication channel to a trusted party. Then, one constructs a reduction that maps real protocol runs to protocol runs in the ideal world, and shows that honest parties cannot distinguish real and ideal protocol executions. The above formulation was introduced by Beaver [5, 3, 4], and extended by Canetti as the universal composability framework [10, 11, 12]. It has also been ported to a modular, formal models-type approach called *reactive systems*, which emphasizes independent analysis of cryptography and communication layers, by Pfitzmann and Waidner [33, 34].

Formal modelling of protocols and cryptographic primitives via real vs. ideal simulations is an increasingly respected paradigm for the analysis of multi-party protocols, including authentication and key-exchange [15, 21, 14], zero-knowledge proofs [13, 16], and the universe of cryptographic primitives [29]. More recently, an RFID privacy-oriented protocol has been proven secure in a strong real/ideal setting [1]. In Section 5, we introduce a first attempt at a comprehensive security model for RFID applications. We demonstrate that the man-in-the-middle attack

to the HB+ protocol is captured in our model and therefore, that HB+ cannot be proven secure in that model. We also argue that the model is a reasonable framework to study the real-world security of RFID applications.

4 Towards Practical Secure Anonymous RFID Authentication

An RFID authentication system has three components: tags T, readers R, and a trusted server $\mathcal{S}$. Tags are wireless transponders: they have no power of their own and respond only when they are in an electromagnetic field. Readers are transceivers and generate such fields, which they use to transmit challenges to tags (via wireless broadcast). There are two types of challenges: multicast and unicast. Multicast challenges are addressed to all tags in the range of a reader, whereas unicast challenges are addressed to specific tags. In our protocols below we consider both types of challenges. However, our multicast challenges are just random strings, and *all* tags in the range of a reader R are challenged with the same random string. This kind of action is *not* usually counted as a communication pass in an authentication protocol.

We shall assume that all honest tags T adhere to the system specifications and the requirements of the authentication protocol. The same applies to the honest readers R, and to the trusted server $\mathcal{S}$. Tags are issued with individual private keys K which they share only with the trusted server $\mathcal{S}$. These keys are used by the tags for authentication. We denote by $\mathcal{K}$ the set of all authorized keys (issued by $\mathcal{S}$).

In our RFID authentication protocols we shall assume that honest readers R and the server $\mathcal{S}$ are linked by a secure communication channel (reliable and authenticated).

4.1 A Provably Secure 1-Pass Optimistic Anonymous RFID Authentication Protocol

In Figure 4 we describe a one-pass protocol that authenticates RFID tags anonymously, and that is *optimistic*, in the sense that when the adversary is not active the cost to both the tag and the server is minimized. In this protocol, each reader R broadcasts a random string r_{sys} obtained from the server $\mathcal{S}$, and updated at regular intervals. All the tags in the range of R use the same r_{sys}, but will combine it with a locally generated string r_{tag}, and send (broadcast) to the reader R the MAC: $h = H_K(r_{sys}, r_{tag})$. Here $H_K(\cdot)$ is a keyed hash $H(K, \cdot)$. T computes the value of local string r_{tag} by taking the MAC of its previous value, stored locally. The server also updates the value r_K in a local key look-table –see Figure 5. From this table, and the value r_{tag} sent by T, the server can find a corresponding key K' and check that the value h is that same as $H_{K'}(r_{sys}, r_{tag})$. If the tag T has not been challenged by an unauthorised reader, the value h will be correct. In this case the cost for both the tag and the server is two MACs. However, if the tag has recently interacted with a malicious reader, the stored values will be

R broadcasts r_{sys}

1. $T \rightarrow R \rightarrow \mathcal{S} : r_{tag}, \;\; h = H_K(r_{sys}, r_{tag})$

T updates $r_{tag} = H_K(r_{tag})$

$\mathcal{S}$ accepts T if:

- $\exists \; (r_{tag}, K) \in L$ s.t. $h = H_K(r_{sys}, r_{tag})$, or
- $\exists \; K \in \mathcal{K}$ s.t. $h = H_K(r_{sys}, r_{tag})$.

$\mathcal{S}$ updates $r_K = H_K(r_{tag})$ in the look-up table L.

Fig. 4. A one-pass anonymous RFID authentication protocol. Figure 5 shows the lookup table L.

strings	r_{K_1}	r_{K_2}	$\cdots$	r_{K_n}
keys	K_1	K_2	$\cdots$	K_n

Fig. 5. The key look-up table L

out-of-sync. Then the server will have to exhaustively search through all keys $k \in \mathcal{K}$ to find the correct value and resynchronize. Note that in the dishonest case the extra computational cost is borne out by the server and not by the tag. By exploiting the higher computational capabilities of the trusted server, we have designed a strong authentication protocol that provably hides the identity of tags from eavesdroppers and malicious readers without requiring the tag to ever perform expensive public-key operations. In all cases, the tag only needs to compute two MACs to authenticate itself. In the honest case, this is also the protocol cost for the central server.

Theorem 3 ([9]). *The 1-pass optimistic anonymous protocol is available, anonymous and secure in the security framework defined below (Section 5).*

To the best of our knowledge, this is the only anonymous, strong RFID authentication protocol that is also amenable to being proven secure within a comprehensive adversarial model, which we describe in the next section.

In the following section we formalize the security definitions for RFID protocols. The model largely follows existing paradigms for security of general-purpose network protocols, but becomes specific to the context of RFID applications in two aspects. First, we consider *availability* explicitly, capturing security against unauthorized disabling of tags directly within the model.

Secondly, we restrict concurrency by prohibiting tags from executing more than one session at a time. Note that this is a restriction only on individual, honest tags—many honest tags can be executing concurrently. In addition, readers (whether honest or corrupt), the central server, and dishonest tags can execute multiple sessions simultaneously. Yet, the requirement that a single honest tag can participate only in one session at a time facilitates the design of concurrently secure protocols. As the restriction is a mild one, and in accordance with the

capabilities of RFID technology, it is beneficial in that it enables designers of security protocols to concentrate on the crucial security aspects and on how to balance competing interests, such as requirements of low computational cost and low memory utilization.

Proof structure. The proof (omitted here) consists of two stages. First, security is shown in an idealized protocol model, wherein honest parties have access to a trusted party (ideal functionality). Security in this ideal world can be readily seen to follow from the behavior of the ideal functionality in simulations. It is then shown that the environment cannot distinguish between real and ideal world simulations. The adversary is allowed to schedule the actions of honest parties, eavesdrop in communications, and interact with the environment in an arbitrary manner (Byzantine adversary).

5 Security Simulation

Initialization of honest parties. Honest parties are initialized as follows. The trusted server—symbolized by oracle $\mathcal{O}_S$—creates a database of keys $K_i, i = 1, \ldots, n$—choosing keys at random from $\{0,1\}^\tau$, where τ is a security parameter (provided as input to the initialization step). For simplicity, we do not consider dynamic corruption of tags here. Instead, the adversary is initialized with a subset of the valid keys $K_{\ell+1}, \ldots, K_n$, and so the first ℓ keys correspond to honest tags. During real-world simulations, the adversary interacts with honest tag T_i by accessing oracle $\mathcal{O}_i$, which emulates the behavior of the honest tag with corresponding key K_i.

The initialization also requires, for each ordered pair (i,j), $1 \leq i < j \leq \ell$, that one chooses two bits $b^1_{i,j}$ and $b^2_{i,j}$, independently at random. For each triple (i,j,c), with $c \in \{1,2\}$, an *ambivalent oracle* $\mathcal{O}^c_{i\vee j}$ will use key K_i or K_j in the simulation, respectively, if $b^c_{i,j} = 0$ or $b^c_{i,j} = 1$. The role of the ambivalent oracles will soon be made clear.

As the simulation starts, each tag oracle or ambivalent oracle is marked as `available`. Each tag oracle or ambivalent oracle independently initializes values r_i, $r^c_{i\vee j}$ at random. The server $\mathcal{O}_S$ generates a random value r^0_{sys} which will be broadcast by readers as challenge to tags during the first *server period*, or simply *period*. Subsequently, the adversary may cause new periods to commence by telling $\mathcal{O}_S$ to refresh the value r^t_{sys} with a new random value, where t counts how many periods have completed before the current one.

Real simulation model. Let $\mathcal{A}$ be the adversary. $\mathcal{A}$ can internally represent many adversarial tags T' (with compromised valid keys or invalid keys) and dishonest readers R', but we represent it as a single party $\mathcal{A}$.

At the beginning of the simulation, the total number of tags n is provided to the adversary. The adversary interacts in an arbitrary manner with the simulation environment $\mathcal{Z}$. Consider a single communication session between $\mathcal{A}$ and some honest party. $\mathcal{Z}$ maintains a notion of time—we do not require synchronized clocks, $\mathcal{Z}$ only needs to discern which adversarial actions precede other

adversarial actions. We now describe what types of messages can be understood by honest parties in the real protocol simulation. We note that, since individual tags execute sequentially, they are not always available to initiate new communication sessions with $\mathcal{A}$, for instance if already communicating with $\mathcal{A}$.

REFRESH(): Called by $\mathcal{A}$ to cause the beginning of a new server period. $\mathcal{O}_S$ increments the period counter ($t \leftarrow t+1$) and generates a new random value r^t_{sys}. This value will be broadcast by honest readers as challenge to tags, until the beginning of the next server period—i.e., until another call to REFRESH() occurs.

START(i): If $\mathcal{O}_i$ is not `available`, this call is ignored. Otherwise $\mathcal{O}_i$ changes status to `communicating` with $\mathcal{A}$, and all oracles of the type $\mathcal{O}^c_{i\vee?}$ are marked as `unavailable`.

START($i \vee j, c$): If $\mathcal{O}^c_{i,j}$ is not `available`, this call is ignored. Otherwise $\mathcal{O}^c_{i,j}$ changes status to `communicating` with $\mathcal{A}$, and $\mathcal{O}_i$, $\mathcal{O}_j$ become `unavailable`, as well as all *other* oracles of the type $\mathcal{O}^{c'}_{i\vee?}$, $\mathcal{O}^{c'}_{j\vee?}$, with $c' \in \{1,2\}$.

SEND(i, m): If $\mathcal{O}_i$ is not `communicating` with $\mathcal{A}$ this call is ignored. Otherwise, $\mathcal{O}_i$ responds with the pair $r_i, h = H_{K_i}(m, r_i)$, and updates $r_i \leftarrow H_{K_i}(r_i)$.

SEND($i \vee j, c, m$): If $\mathcal{O}^c_{i\vee j}$ is not `communicating` with $\mathcal{A}$ this call is ignored. Otherwise, let ι be either i or j, corresponding to whether $\mathcal{O}^c_{i\vee j}$ was initialized with key K_i or K_j, respectively. Then $\mathcal{O}^c_{i\vee j}$ responds with the pair $r^c_{i\vee j}, h = H_{K_\iota}(m, r^c_{i\vee j})$, and updates $r^c_{i\vee j} \leftarrow H_{K_\iota}(r^c_{i\vee j})$.

SEND($\mathcal{TS}, m$): $\mathcal{O}_S$ parses m as a string $r||h$. It then consults its lookup table for an entry of the type (r, K_i). If such an entry is found, $\mathcal{O}_S$ further checks if $h = H_{K_i}(r^t_{sys}||r)$, replying to $\mathcal{A}$ with 1 (indicating authentication success) if the equality holds. If either a match is not found or the check fails, $\mathcal{O}_S$ searches its key database $\mathcal{K}$ for any K_i such that $h = H_{K_i}(r^t_{sys}||r)$. If such K_i is found, it replies to $\mathcal{A}$ with 1, or 0 otherwise. $\mathcal{O}_S$ outputs the identity i if the authentication is successful with key K_i, else it outputs nothing. This output is not observed by the environment $\mathcal{Z}$.

END(i): If $\mathcal{O}_i$ is not `communicating`, the call is ignored. Otherwise, $\mathcal{O}_i$ becomes `available`, as well as any $\mathcal{O}^c_{i\vee j}$ such that $\mathcal{O}_j$ is also `available`.

END($i \vee j, c$): If $\mathcal{O}^c_{i\vee j}$ is not `communicating`, the call is ignored. Otherwise, $\mathcal{O}^c_{i\vee j}$ becomes `available`, as well as $\mathcal{O}_i$, $\mathcal{O}_j$, and any $\mathcal{O}^{c'}_{i\vee?}$, $\mathcal{O}^{c'}_{j\vee?}$, for $c' \in \{1,2\}$.

The role of the identity-ambivalent oracles. The ambivalent oracles $\mathcal{O}^c_{i\vee j}$ enable $\mathcal{A}$ to interact with parties whose identity is one of two possible choices. This enables attacks against anonymity, where $\mathcal{A}$'s objective is to determine if $\mathcal{O}^1_{i\vee j}$ and $\mathcal{O}^2_{i\vee j}$ represent the same or different identities. Note that the concurrency-prevention rules (enforced via the tags maintaining a status among `available`, `communicating`, and `unavailable`) are designed to prevent that $\mathcal{A}$ may disambiguate the ambivalent oracles simply on the basis of availability conflicts, while at the same time preventing that a single tag executes two sessions concurrently.

5.1 Security Definitions

We now formally define the security goals of anonymous authentication protocols. We define a session *ses* with honest tag T_i as a time-interval between the first call to $\mathsf{START}(i)$ after either the beginning of the simulation or the most recent call to $\mathsf{END}(i)$, and the first subsequent call to $\mathsf{END}(i)$.

Availability: holds when there is no efficient adversary $\mathcal{A}$ that during the course of the simulation, has non-negligible probability in preventing a tag T_i from authenticating itself to a reader R_j during a session *ses*, without changing T_i's interaction with R_j in session *ses*. This should remain true even if $\mathcal{A}$ has interacted with T_i or $\mathcal{TS}$ arbitrarily in the past, perhaps attempting to force either or both into an inconsistent state. Note that $\mathcal{A}$ is still allowed to interact with all other honesty parties, including reader R_j, during *ses*. The advantage $adv^{\mathcal{A},T_i}_{AVLB}$ of $\mathcal{A}$ in this game against T_i is the maximum probability that $\mathcal{TS}$ rejects T_i in any session.

$$adv^{\mathcal{A},T_i}_{AVLB} := Prob\left[\mathcal{TS} \text{ rejects } T_i \text{ in } ses \mid \mathcal{A} \text{ only relays between } \mathcal{O}_i \text{ and } R_j \text{ during } ses\right],$$

and $adv^{\mathcal{A}}_{AVLB}$ is defined as the maximum of the $adv^{\mathcal{A},T_i}_{AVLB}$, over all honest tags T_i in any session.

An important concern in regard to the management of RFIDs is to have a kill process, in which a reader can instruct an RFID tag to disable its functionality permanently. Current methods for disabling EPC tags have been recently shown ([35]) to allow an attacker to perform a power-analysis based recovery of the kill-key. Such attacks violate the above definition of availability. Our protocols can be adapted to support a kill-key while still guaranteeing availability.

Authentication: holds when there is no efficient adversary that, during the simulation, succeeds with non-negligible probability in authenticating itself to an honest reader R_j during some session *ses*, and moreover: (*a*) The server $\mathcal{TS}$ believes $\mathcal{A}$ to have authenticated itself as tag T_i in *ses*; and (*b*) the duration interval [start-time, end-time] for session *ses* is disjoint from the duration intervals of all of $\mathcal{A}$'s sessions with oracle $\mathcal{O}_i$ as well as with any ambivalent oracle $\mathcal{O}^c_{i,j}$ that was initialized as $\mathcal{O}_i$. We note that in this definition, $\mathcal{A}$ is not required to know under which identity T_i it has succeeded in authenticating itself. Furthermore, it accommodates man-in-the-middle attacks, as long as the attack leads to $\mathcal{A}$'s acquiring knowledge (such as keys) that can be used for subsequent authentication attempts, while ruling out scenarios in which the adversary simply relays messages between honest parties as successful attacks. The advantage $adv^{\mathcal{A},T_i}_{\mathsf{AUTH}}$ of the adversary against authentication is simply the probability that it succeeds.

$$adv^{\mathcal{A},T_i}_{\mathsf{AUTH}} := Prob\left[\mathcal{A} \text{ authenticates as } T_i \text{ in } ses;\ \ ses \cap \text{Sessions}(\mathcal{A}, \mathcal{O}_i) = \emptyset\right],$$

where i is the index of an honest user. The advantage $adv^{\mathcal{A}}_{\mathsf{AUTH}}$ it the maximum of the $adv^{\mathcal{A},T_i}_{\mathsf{AUTH}}$ over all tags T_i.

Anonymity holds when no efficient adversaries have non-negligibly better-than-even chances of, at any time in the simulation, outputting a triple (i, j, b), where $1 \leq i < j \leq n$, and either (1) $b = 0$ and $\mathcal{O}^1_{i \vee j} \neq \mathcal{O}^2_{i \vee j}$, or (2) $b = 1$ and $\mathcal{O}^1_{i \vee j} = \mathcal{O}^2_{i \vee j}$. The advantage of the adversary in distinguishing T_i and T_j, $Adv^{\mathcal{A}, i \vee j}_{\mathsf{ANON}}$, is defined as the difference between winning and losing probabilities when the adversarial guess bit equals 1:

$$adv^{\mathcal{A}, i \vee j}_{\mathsf{ANON}} := Prob\left[(i, j, 1) \leftarrow \mathcal{A} \mid \mathcal{O}^1_{i \vee j} = \mathcal{O}^2_{i \vee j}\right] - Prob\left[(i, j, 1) \leftarrow \mathcal{A} \mid \mathcal{O}^1_{i \vee j} \neq \mathcal{O}^2_{i \vee j}\right],$$

and the adversarial advantage against anonymity, $adv^{\mathcal{A}}_{\mathsf{ANON}}$ is the maximum of the $adv^{\mathcal{A}, i \vee j}_{\mathsf{ANON}}$ over all pairs (i, j), with $i < j$.

This is a unified framework because the adversary does not need to identify, at any particular point in the simulation, which security property it seeks to defeat. Instead, it may weigh its knowledge and adjust its strategy during the simulation to maximize its success in violating any of the security requirements.

References

1. G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable RFID tags via insubvertible encryption. In *Proc. of the ACM Conf. on Computer and Communication Security (ACM CCS 2005)*, pp. 92–101, ACM Press, 2005.
2. B. Awerbuch, D. Holmer, C. Nita-Rotaru and H. Rubens, *An On-Demand Secure Routing Protocol Resilient to Byzantine Failures*, ACM Workshop on Wireless Security – WiSe'02 2002.
3. D. Beaver, *Foundations of secure interactive computing,* Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 377-391, 1991.
4. D. Beaver. Secure multi-party protocols and zero-knowledge proof systems tolerating a faulty minority. In *Journal of Cryptology*, vol. 4, no. 2, pp. 75122, 1991.
5. D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *Proc. of Advances in Cryptology (CRYPTO 89)*, LNCS Vol. 435, pp. 589–590, Springer-Verlag, 1989.
6. E.M. Belding-Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. In *IEEE Personal Communications Magazine*, pp. 46-55, 1991.
7. M. Burmester and T. van Le. Secure Multipath Communication in Mobile Ad hoc Networks. In *Proc. International Conference on Information Technology Coding and Computing*, pp. 405–409, 2004.
8. M. Burmester, T. Van Le, and A. Yasinsac. Adaptive gossip protocols: managing security and redundancy in dense ad hoc networks. In *Journal of Ad hoc Networks*, vol. 4, no. 3, pp. 504–515, Elsevier, 2006.
9. C. Chatmon, T. Le Van, and M. Burmester. Anonymous authentication with RFID devices. *FSU Technical Report: TR-060112.* Available at *url* http://www.sait.fsu.edu/research/rfid/index.shtml.
10. R. Canetti. Studies in Secure Multiparty Computation and Applications. *Ph. D. thesis*, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.
11. R. Canetti. Security and composition of multi-party cryptographic protocols. In *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.

12. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. of Foundations of Comp. Sci. (FOCS 2001)*, pp. 136–145, 2001.
13. R. Canetti and M. Fischlin. Universally Composable Commitments. In *Proc. of Advances in Cryptology (CRYPTO 2001)*, LNCS 2139, pp. 19-ff., Springer-Verlag, 2001.
14. R. Canetti and J. Herzog. Universally Composable Symbolic Analysis of Cryptographic Protocols (The case of encryption-based mutual authentication and key exchange). In *E-print Technical Report # 2004/334*, International Association for Cryptological Research, 2004. Available at *url* http://eprint.iacr.org/2004/334
15. R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels (Extended Abstract). In *Proc. of Advances in Cryptology (EUROCRYPT 2002)*, LNCS 2332, pp. 337–ff., Springer-Verlag, 2002.
16. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *Proc. of the ACM Symposim on Theory of Computing*, vol. 34, pp. 494–503, ACM Press, 2002.
17. J. R. Douceur. The Sybil attack. In *Proc. 1st International Workshop on Peer-to-Peer Systems – IPTPS '02*, 2002.
18. L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, *Princeton*, NJ, 1962.
19. H. Gilbert, M. Rodshaw, and H. Sibert. An Active Attack Against HB+ – A Provably Secure Lightweight Authentication Protocol. *PerSec '04*, March 2004. Full paper available in *E-print Technical Report # 2005/237*, International Association for Cryptological Research, Available at *url* http://eprint.iacr.org/2005/237.pdf
20. M. Hirt and U. Maurer. Player Simulation and General Adversary Structures in Perfect Multiparty Computation. In *Journal of Cryptology*, Vol. 13, No. 1, pp. 31–60, 2000.
21. D. Hofheinz, J. Müller-Quade, and R. Steinwandt. Initiator-Resilient Universally Composable Key Exchange. In *Proc. of the European Symp. on Research in Computer Security (ESORICS 2003)*, LNCS 2808, pp. 61–84, Springer-Verlag, 2003.
22. N. J. Hopper and M. Blum. Secure Human Identification Protocols. In *Proc. of Advances in Crypotology (ASIACRYPT 2001)*, LNCS, Springer-Verlag, 2001.
23. Y-C Hu, D.B. Johnson and A. Perrig. Ariadne: A Secure On-Demand Routing protocol for Ad Hoc Networks. In *Proc. of the ACM Annual Intern. Conf. on Mobile Computing and Networking (MobiCom 2002)*, ACM Press, 2002.
24. Y-C Hu, D.B. Johnson and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. In *Proc. 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002)*, IEEE, Calicoon, NY, 2002.
25. Y-C. Hu, A. Perrig and D.B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proc. of WiSe2003*, pp. 30–40, 2003.
26. D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad-Hoc Wireless Networks. In *ed. T. Imielinski and H. Korth, Mobile Computing*, Kluwer Academic Publisher, pp. 152–181, 1996.
27. A. Juels and S. A. Weiss. Authenticating Pervasive Devices with Human Protocols. In *Proc. of Advances in Cryptology—CRYPTO 2005*, LNCS vol. 3621, pp. 293–ff, Springer-Verlag, 2005.
28. J. Katz and J. S. Shin. Parallel and Concurrent Security of the HB and HB+ Protocols. To appear in *Proc. of Advances in Cryptology (EUROCRYPT 2006)*, Springer, 2006.

29. P. Laud. Formal analysis of crypto protocols: Secrecy types for a simulatable cryptographic library. In *Proc. of the 12th ACM Conf. on Computer and Communications Security (ACM CCS 2005)*, pp. 26–35, ACM Press, 2005.
30. P. Papadimitratos and Z.H. Haas. Secure Routing for Mobile Ad hoc Networks. In *Mobile Computing and Communications Review*, Vol 6, No 4, 2002.
31. C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing for Mobile Computers. In *Computer Communications Review*, pp. 224-244, 1994.
32. C.E. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, 1999.
33. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. of the ACM Conf. on Computer and Communications Security (ACM CCS 2000*, pp. 245–254, ACM Press, 2000.
34. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. of the IEEE Security and Privacy Symposium (S & P 2001)*, pp. 184–200, 2001.
35. Y. Oren and A. Shamir. Power Analysis of RFID Tags. *Invited talk, RSA Conference, Cryptographer's Track (RSA-CT 2006).* Available at `http://www.wisdom.weizmann.ac.il/~yossio/rfid`
36. A.J. Menezes, P.C. van Oorschot and S.A. Vanscott. *Handbook of Applied Cryptography*, CRC Press, 1996.
37. G. Tsudik. YA-TRAP: Yet another trivial rfid authentication protocol. *International Conference on Pervasive Computing and Communications*, 2006.
38. M. G. Zapata. Secure Ad hoc On-Demand Vector (SAODV) Routing. *IETF Internet Draft.* Available at *url* http://www.potaroo.net/ietf/all-ids/draft-guerrero-manet-saodv-00.txt (Work in Progress).

Oblivious Scalar-Product Protocols

Huafei Zhu and Feng Bao

Institute for Infocomm Research, A-star, Singapore
{huafei, baofeng}@i2r.a-star.edu.sg

Abstract. In this paper, a new notion which we call oblivious scalar-product protocols is introduced and formalized. We then propose an efficient implementation of oblivious scalar-product protocols based on homomorphic cryptographic primitives (e.g., homomorphic encryptions and homomorphic commitments). Finally we show that our implementation is provably secure assuming that the underlying Fujisaki-Okamoto's commitment scheme is statistically hiding and computationally binding, and Paillier's encryption scheme is semantically secure in the common reference string model.

Keywords: Oblivious scalar-product protocol, homomorphic commitment scheme, homomorphic public key encryption.

1 Introduction

Suppose that Alice who has an input vector $inp_A = (1, x_1, \cdots, x_l)$ ($x_i \in F$, $1 \leq i \leq l$, F is a finite field) and Bob who has an input vector $inp_B = (y_0, y_1, \cdots, y_l)$ ($y_i \in F$, $0 \leq i \leq l$), wish to compute scalar-product obliviously so that at the end of the protocol, Alice learns $\sum_{i=0}^{l} x_i y_i$ while Bob learns nothing. We call this an oblivious scalar-product protocol.

The structure of oblivious scalar-product protocols informally defined above is asymmetric (Alice's first input is always 1 while Bob's first input is an arbitrary element $y_0 \in F$). If we assume that $y_0 \equiv 0$, then the structure of oblivious scalar-product protocols is reduced to the symmetric form where Alice's input vector inp_A is $(x_1, \cdots, x_l)$ while Bob's input vector inp_B is $(y_1, \cdots, y_l)$ and the output of Alice is $\sum_{i=1}^{l} x_i y_i$ − a standard output of a scalar-product protocol. This paper however, insists on efficient implementations of asymmetric forms since such asymmetric scalar-product protocols are certainly welcome in certain cryptographic scenarios (see section 1.3 for more details).

We stress that the notion of oblivious scalar-product protocols is different from the notion of shared-scalar-product protocols. In the later case (say [18]), there are two participants Alice who holds her input vector $(x_1, \cdots, x_l)$ ($x_i \in F$, $1 \leq i \leq$ l, where F is a finite field) and Bob who holds his input vector $(y_1, \cdots, y_l)$ ($y_i \in F$, $1 \leq i \leq$ l)). They wish to compute random shares of scalar product $s_a \in F$ and $s_b \in F$ such that $\sum_{i=1}^{l} x_i y_i = s_a + s_b$. At the end of the protocol, Alice holds the value s_a while Bob holds s_b.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 313–323, 2006.
© Springer-Verlag Berlin Heidelberg 2006

1.1 Related Work

The notion of oblivious scalar-product protocols is closely related to the notion of oblivious polynomial evaluation (OPE). The later was first introduced and formalized by Naor and Pinkas at STOC'99 [13]. Informally, an oblivious polynomial evaluation involves two parties (say, Alice and Bob). Alice's input is an element $\alpha \in F$, and Bob's input is a polynomial $f(x)$ of degree l ($l \geq 1$, i.e., $f(\alpha)$=$a_0 + a_1\alpha + \cdots + a_l\alpha^l \in F$). At the end of execution of the protocol the Alice learns $f(\alpha)$ and Bob learns nothing.

Now, we if rewrite a polynomial $f(x) = a_0 + a_1x + \cdots + a_lx^l$ in the context of oblivious polynomial evaluation as a scalar-product $(a_0, a_1, \cdots, a_l) \cdot (x^0, x^1, \cdots, x^l)$ $(= f(x)$, in the context of oblivious scalar-product). Clearly, an input of Alice in an oblivious scalar-product protocol is a vector $(x_0, x_1, \cdots, x_l)$, where $x_i \in F$ while an input of Alice in an oblivious polynomial evaluation protocol is a value $x \in F$ from which an input vector of the special form $(x^0, x^1, \cdots, x^l)$ can be induced. It follows that an input vector $(x^0, x^1, \cdots, x^l)$ is a special form of a general input $(x_0, x_1, \cdots, x_l)$. As a result, the notion of oblivious scalar-product protocols is a general case of the notion of oblivious polynomial evaluation protocols.

1.2 This Work

In this paper, a new notion called oblivious scalar-product protocols is introduced and formalized and then an efficient implementation of oblivious scalar-product protocols is proposed. The security definition of oblivious scalar-product protocols is defined in terms of the general ideal-vs-real framework. We stress that the rewinding of the malicious party is allowed in our model since a malicious party is not allowed to communicate with distinguisher D, i.e., the distinguisher D only gets to see the transcript to protocol execution which is significant difference from the argument of the universally composable property [4] where the rewinding of a malicious party is strictly forbidden in Canetti's model [4]. However, we do not deal with the security of our protocol in Canetti's model throughout the paper.

The development of the techniques for general secure multiparty computation showed that any two-party function can be computed securely[1, 3, 6, 10, 16]. Thus, Alice and Bob can perform oblivious scalar-product protocol even without help from a third party. However, even for relatively simple functions, this may be prohibitively expensive since the number of cryptographic operations performed is proportional to the size of the circuit computing $f(x)$. Therefore it is interesting to implement oblivious scalar-product protocol that does not emulate the circuit for the function. With this goal in mind, we propose an efficient implementation of oblivious scalar-product protocols from homomorphic cryptographic primitives. Finally, we are able to show that our implementation is provably secure in the common reference model. That is,

Claim 1: For any malicious Alice A, there exists a simulator sim_A that plays the role of A in the ideal world such that for any polynomial time distinguisher D, the view of D when it interacts with A in real conversation is

computationally indistinguishable from that when it interacts with sim_A in the ideal world.

Claim 2: For any malicious Bob B, there exists a simulator sim_B that plays the role of B in the ideal world such that for any probabilistic polynomial time distinguisher D, the view of D when it interacts with B in real conversation is computationally indistinguishable from that when it interacts with sim_B in the ideal world.

It follows that our implementation is secure against static adversary who corrupts Alice or Bob in the common reference string model assuming that the underlying Fujisaki-Okamoto's commitment scheme is statistically hiding and computationally binding, and Paillier's encryption scheme is semantically secure.

1.3 Applications

Like its sibling notion − oblivious polynomial evaluation, the potential areas of applications of our primitive are numerous (e.g., secret-key zero-knowledge, secure computation of scar-product protocols, and so on....). We thus provide the following two illustrative applications of the primitive.

Secret-key zero-knowledge setup protocols. It is easy to see that when $l = 1$, an implementation of our oblivious scalar-product protocols implies an implementation of key-setup protocols for secret-key zero-knowledge proof systems [5]. Thus our notion can be viewed as a general extension of key-setup protocols for secret-key zero-knowledge proof systems as well.

Scalar-product protocols and shared-scalar-product protocols. Privacy preserving data mining is a new and rapidly emerging research area, where data mining algorithms are analyzed for the side-effects they incur in data privacy. Secure shared scalar product protocols are the fundamental cryptographic build blocks for building secure data mining protocols. Recall that a shared scalar product protocol is the following thing: there are two participants Alice whose input is a vector $(x_1, \cdots, x_l)$ $(x_i \in F)$and Bob whose input is a vector $(y_1, \cdots, y_l)$ $(y_i \in F)$. They wish to compute random shares of scalar product $s_a \in F$ and $s_b \in F$ such that $\sum_{i=1}^{l} x_i y_i = s_a + s_b$. At the end of the protocol, Alice holds the value s_a while Bob holds s_b.

Several private shared scalar-product protocols have been proposed in the context of privacy-preserving data ming. In [11], Goethals et al have already shown that several shared scalar product protocols (e.g., those presented in [8] and [15]) are insecure even in the semi-honest model. Goethals et al further proposed an interesting solution to the shared-scalar-product protocols based on the homomorphic encryption scheme (*vic.*, Paillier's encryption scheme [14]) and showed that their construction is provably secure in the semi-honest model. Very recently, Zhu et al [18] proposed an efficient implementation of shared-scalar-protocols based on the homomorphic cryptographic primitives which is provably secure assuming that the underling homomorphic encryption scheme

is semantically secure and the homomorphic commitment scheme is statistically hiding and computationally binding in the public reference string model.

It is not hard to see that an efficient implementation of oblivious scalar-product protocols implies an efficient implementation of shared scalar product protocols. That is, given an implementation of oblivious scalar-product protocol, we can evaluate an shared-scalar-product protocol trivially by setting $s_b \leftarrow (-a_0 \in F)$, and $s_a \leftarrow f(x) = a_0 + a_1 x + \cdots + a_l x^l$. This can be verified as follows: $(a_1, \cdots, a_l) \cdot (x, \cdots, x^l) = a_1 x + \cdots + a_l x^l = (-a_0) + a_0 + a_1 x + \cdots + a_l x^l = (-a_0) + f(x)$.

Comparison. At FC'05, Kiayias and Mitrofanova presented three protocols for secure computation of scalar-product evaluation [12]. To the best of our knowledge, this is first construction of scalar-product protocols in the malicious model from homomorphic cryptographic primitives. We stress that all three schemes presented in [12] are constructed from ElGamal-like homomorphic encryptions, where the underlying homomorphic encryption scheme used in their construction is of special form (i.e., $(g^r, h^r f^m)$), where g is an element of order q in Z_p^*, $p = 2q + 1$, $h, f \in < g >$). It follows that Kiayias and Mitrofanova's homomorphic encryption allows one to encrypt a short message (e.g., a bit $b \in \{0, 1\}$). Thus, the inputs of scalar-product protocols proposed in [12] are restricted to any finite field of small sizes. This restriction is crucial for their implementation, and thus it cannot be removed from their constructions. Our scalar-product protocols can work over general finite fields. This is the most significant feature of our protocols different from that of Kiayias and Mitrofanova's.

Road map. The rest of paper is organized as follows: In Section 2, syntax, functionality, and security definition for oblivious scalar-product protocols are presented. In Section 3, building blocks that will be used to implement oblivious scalar-product protocols are sketched and an implementation and the proof of security are proposed. We conclude our work in Section 4.

2 Syntax, Functionality, and Security Definition

2.1 Syntax

An oblivious scalar-product protocol consists of the following two probabilistic polynomial time (PPT) Turing machines:

- On input system parameter l, a PPT Turing machine A (say, Alice), chooses l elements $x_1, \cdots, x_l \in F$ uniformly at random (throughout the paper, we assume that $F = Z_m^*$, where m is a large prime number). The input vector of Alice is denoted by $inp_A = (1, x_1, \cdots, x_l)$;
- On input system parameter and l, a PPT Turing machine B (say, Bob), chooses $(l + 1)$ elements $y_0, y_1, \cdots, y_l \in Z_m$ uniformly at random. The input vector of Bob is denoted by $inp_B = (y_0, \cdots, y_l)$;
- On inputs inp_A and inp_B, Alice and Bob jointly compute the value $\sum_{i=0}^{l} x_i y_i \mod m$;
- The output of Alice is $\sum_{i=0}^{l} x_i y_i \mod m$ while the output of Bob is $\perp$.

2.2 Functionality

The functionality $\mathcal{F}_{OSP}$ of oblivious scalar-product protocol (OSP) can be abstracted as follows:

- A player (say Alice) has her input vector inp_A=$(1, x_1, \cdots, x_l)$; Another player (say Bob) has his input vector inp_B=$(y_0, y_1, \cdots, y_l)$; Each participant sends the corresponding input set to $\mathcal{F}_{OSP}$ – an imaginary trusted third party in the ideal world via a secure and private channel.
- Upon receiving inp_A and inp_B, $\mathcal{F}_{OSP}$ checks whether $x_i \in Z_m$ and $y_i \in Z_m$ $(1 \leq i \leq l)$.
  If the conditions are satisfied, then $\mathcal{F}_{OSP}$ computes $\sum_{i=0}^{l} x_i y_i \bmod m$;
  If there exists a subset $\tilde{s_A} \subset inp_A$ such that each $x_i \in \tilde{s_A}$ but $x_i \notin Z_m$, then $\mathcal{F}_{OSP}$ chooses an element $x_i' \in_r Z_m$ and substitutes x_i with x_i'. Similarly, if there exists a subset $\tilde{s_B} \subset inp_B$ such that each $y_i \in \tilde{s_B}$ but $y_i \notin Z_m$, then $\mathcal{F}_{OSP}$ chooses an element $y_i' \in_r Z_m$ and substitutes y_i with y_i'. By inp_A=$(x_1, \cdots, x_l)$ (using the same notation of the input vector of Alice) we denote the valid input set of Alice which may be modified by $\mathcal{F}_{OSP}$ and by inp_B=$(y_1, \cdots, y_l)$ (again using the same notation of the input vector of Bob), we denote the valid input set of Bob which may be modified of the original input values by $\mathcal{F}_{OSP}$. Once given the valid input sets inp_A and inp_B, $\mathcal{F}_{OSP}$ computes $\sum_{i=0}^{l} x_i y_i \bmod m$.
  Finally, $\mathcal{F}_{OSP}$ sends $\sum_{i=0}^{l} x_i y_i \bmod m$ to Alice via the secure and private channel while Bob learns nothing.
- The output of Alice is $\sum_{i=0}^{l} x_i y_i \bmod m$ which is sent by $\mathcal{F}_{OSP}$ via the secure and private channel between them. The output of Bob is $\perp$.

2.3 Security Definition

The security definition of oblivious scalar-product protocols is defined in terms of the general ideal-vs-real framework. In this framework, we first consider an ideal model in which two real participants are joined by a third trusted party, and the performance is via this trusted party. Next, we consider the real model in which a real two-party protocol is executed. A protocol in the real model is said to be secure with respect to certain adversarial behavior if the possible real execution with such an adversary can be simulated in the ideal model. That is, we want to show that there exists a polynomial time computable transform of adversarial behavior in the real conversation into corresponding adversarial behavior in the ideal model.

Definition 1. *An oblivious scalar-product protocol is secure for Alice A if for any malicious Bob B, there exists a simulator sim_B that plays the role of B in the ideal world such that for any probabilistic polynomial time distinguisher D, the view of D when it interacts with B in real conversation is computationally indistinguishable from that when it interacts with sim_B in the ideal world.*

Definition 2. *An oblivious scalar-product protocol is secure for Bob B if for any malicious Alice A, there exists a simulator sim_A that plays the role of A in the ideal world such that for any probabilistic polynomial time distinguisher D, the view of D when it interacts with A in real conversation is computationally indistinguishable from that when it interacts with sim_A in the ideal world.*

Definition 3. *An oblivious scalar-product protocol is secure for any static probabilistic polynomial time (PPT) adversary if it is secure for any PPT Alice and any PPT Bob.*

3 The Construction and Proof of Security

We propose an efficient implementation of OSP protocols based on the homomorphic primitives. We then show that our construction is provably secure in the common reference string model assuming that Paillier's encryption scheme is semantically secure and Fujisaki-Okamoto's commitment scheme is statistically hiding and computationally binding.

3.1 Homomorphic Cryptosystem

The implementation of oblivious scalar-product protocols is based on the homomorphic cryptographic primitives (homomorphic encryption scheme, and homomorphic commitment scheme), we therefore briefly review these building blocks below:

Paillier's public key encryption scheme. Paillier investigated a novel computational problem called the composite residuosity class problem (CRS), and its applications to public key cryptography in [14] sketched below.

The public key is a k_1-bit RSA modulus $n = pq$, where p, q are two large safe primes. The plain-text space is Z_n and the cipher-text space is $Z^*_{n^2}$. To encrypt $a \in Z_n$, one chooses $r \in Z^*_n$ uniformly at random and computes the cipher-text as $E_{PK}(a, r) = g^a r^n \bmod n^2$, where $g = (1 + n)$ has order n in $Z^*_{n^2}$. The private key is (p, q). Given a cipher-text $c{:=}g^a r^n \bmod n^2$, the decryption algorithm computes $a \in Z_n$ from the following equation $\frac{L(c^\lambda \bmod n^2)}{L((1+n)^\lambda \bmod n^2)} \bmod n$. Paillier's public key encryption scheme is homomorphic, i.e., $E_{PK}(a_1, r_1) \times E_{PK}(a_2, r_2) \bmod n^2 = E_{PK}(a_1 + a_2 \bmod n,\, r_1 \times r_2 \bmod n)$.

Fujisaki-Okamoto commitment scheme. Let s be a security parameter. The public key is a k_2-bit RSA modulus, where P, Q are two large safe primes. We assume that neither the committer C nor the receiver R knows factorization N. Let g_1 be a generator of QR_N and g_2 be an element of large order of the group generated by g_1 such that both discrete logarithm of g_1 in base g_2 and the discrete logarithm of g_2 in base g_1 are unknown by C and R. We denote $C(a, r_a) = g_1^a g_2^{r_a} \bmod N$ a commitment to x in base (g_1, g_2), where r_a is randomly selected over $\{0, 2^s N\}$. This commitment scheme first appeared in [9] and reconsidered by Damgård and Fujisaki [7] is statistically secure commitment scheme, i.e., 1) C is

unable to commit itself to two values a_1, a_2 such that $a_1 \neq a_2$ in $\mathcal{Z}$ by the same commitment unless R can factor N or solves the discrete logarithm of g_1 in base g_2 or the the discrete logarithm of g_2 in base g_1; and 2) $C(a, r_a)$ statistically reveals no information to R, *i.e.*, there is a simulator which outputs simulated commitments to a which are statistically indistinguishable from true ones.

Notice that this commitment is homomorphic, i.e., $C(a+b, r_a+r_b) = C(a, r_a) \times C(b, r_b)$. This property is useful when R wants to prove that the committed value $a \in [x, y]$.

3.2 Our Implementation

The technique used to implement our primitive is closely related to what have been proposed in [17, 18]. We stress that the proof of security presented in paper however is significant difference from that presented in [18] where Bob is allowed to choose and commit salts for his final output of shared scalar-product protocols but Bob is not allowed to obtain anything in the oblivious scalar-product protocols. The implementation of oblivious scalar-product protocols consists of two stages: pre-processing stage and secure two-party computation stage.

Pre-processing stage. In the pre-processing stage, the following required computation are proceeded: a public key of the commitment scheme is a k_2-bit RSA modulus, where P, Q are two large safe primes ($|P| = |Q| = k_2/2$). We assume that neither A nor B knows factorization N ($N = P\ Q$). Let g_1 be a generator of QR_N and g_2 be an element of large order of the group generated by g_1 such that both discrete logarithm of g_1 in base g_2 and the discrete logarithm of g_2 in base g_1 are unknown by A and B. The public reference string σ contains (g_1, g_2, N) as well as other related strings for description of group structure. We assume that σ is provided by an oracle (thus our model for implementation of oblivious scalar-product protocols is within the common reference string model, we however remark that the use of Fujisaki-Okamoto commitment scheme is not essential, it can be replaced by any homomorphic commitment scheme).

Secure computation stage. In the secure computation stage, Alice and Bob involve the following performances:

Step 1. On input $inp_A = (1, x_1, \cdots, x_l)$, Alice commits each element $x_i \in inp_A$ using Fujisaki-Okamoto commitment scheme. By $C(x_i, r_{x_i}) = g_1^{x_i} g_2^{r_{x_i}} \bmod N$, we denote the commitment of x_i with the random string r_{x_i}.

On input a security parameter k_1, Alice generates a pair of public key pk_A and security key sk_A for Paillier public key encryption scheme and the RSA modulus $n = pq$ is of size k_1 which is sufficiently large so that all computations can be taken over $\mathcal{Z}$.

Alice encrypts x_i by means of the Paillier's encryption scheme. By $E_A(x_i)$, we denote the cipher-text of x_i under Alice's public key pk_A.

Alice sends $E_A(x_i)$ and $C(x_i, r_{x_i})$ $(1 \leq i \leq l)$ to Bob and proves to Bob that the encryption $E_A(x_i)$ and the commitment $C(x_i, r_{x_i})$ hide the same value.

Also Alice proves to Bob that the corresponding committed value x_i lies in the correct interval Z_m (=$[1, m]$) using Boudot's protocol [2];

Step 2. On input inp_B=$(y_0, y_1, \cdots, y_l)$, Bob commits each element $y_i \in inp_B$ using Fujisaki-Okamoto commitment scheme. By $C(y_i, r_{y_i}) = g_1^{y_i} g_2^{r_{y_i}} \bmod N$, we denote the commitment of y_i with the random string r_{y_i}.
Bob also chooses $\tilde{y_0}, \tilde{y_1}, \cdots, \tilde{y_l} \in_r I$ uniformly at random (where $I = \{0,1\}^{|m|+k'}$, e.g., k'=160 bit), and performs the following computations:

$$C(\tilde{y_0}, r_{\tilde{y_0}}), C(\tilde{y_1}, r_{\tilde{y_1}}), \cdots, C(\tilde{y_l}, r_{\tilde{y_l}})$$

$$E_A(z) = E_A(1)^{y_0} E_A(x_1)^{y_1} \cdots E_A(x_l)^{y_l}$$

$$E_A(\tilde{z}) = E_A(1)^{\tilde{y_0}} E_A(x_1)^{\tilde{y_1}} \cdots E_A(x_l)^{\tilde{y_l}}$$

Finally, Bob sends $E_A(z)$, $E_A(\tilde{z})$, together with the proof that he knows how to open the commitments $C(y_i, r_{y_i})$, $C(\tilde{y_i}, r_{\tilde{y_i}})$ and $y_i \in Z_m$ and $\tilde{y_i} \in I$ are in the correct interval $(0 \le i \le l)$.

Step 3. Alice verifies the correctness of the proof. If it is incorrect, then she stops the execution of the protocol; otherwise, she performs the following computations:

$$z = D_A(E_A(z)), \tilde{z} = D_A(E_A(\tilde{z}))$$

Finally, Alice chooses a random string $f \in \{0,1\}^{k_4}$ (e.g., k_4 =160-bit) uniformly at random and sends it to Bob;

Step 4. Both parties perform the following computations:

$$C(z_0) \leftarrow C(y_0, r_{y_0})^f C(\tilde{y_0}, r_{\tilde{y_0}})$$

$$C(z_1) \leftarrow C(y_1, r_{y_1})^f C(\tilde{y_1}, r_{\tilde{y_1}})$$

$$\cdots$$

$$C(z_l) \leftarrow C(y_l, r_{y_l})^f C(\tilde{y_l}, r_{\tilde{y_l}})$$

Step 5. Bob opens $C(z_0)$, $\cdots$, $C(z_l)$ to Alice;

Step 6. Alice checks correctness of the opening of commitments, and also checks the validity of equation below:

$$\tilde{z} + fz = (z_0, z_1, \cdots, z_l) \cdot (1, x_1, \cdots, x_l)$$

Step 7. If all are correct, Alice outputs z; otherwise outputs $\perp$.

3.3 The Proof of Security

Lemma 1. *For any malicious Alice A, there exists a simulator sim_A that plays the role of A in the ideal world such that for any polynomial time distinguisher D, the view of D when it interacts with A in real conversation is computationally indistinguishable from that when it interacts with sim_A in the ideal world.*

Proof: sim_A first generates system parameters − the public key of the underlying commitment scheme is (N, g_1, g_2), the secret key (trapdoor of the commitment scheme) is (P, Q, w), where $N = PQ$, $g_2 = g_1^w$, $g_1 \in QR_N$ is a public reference string. Unlike the real protocol, sim_A is allowed to hold the auxiliary information of the underlying commitment scheme.

sim_A first rewinds A to extract $x_i \in Z_m$ from the zero-knowledge proof of knowledge in Step 1 and gives the input vector $inp_A = (1, x_1, \cdots, x_l)$ to $\mathcal{F}_{OSP}$.

$\mathcal{F}_{OSP}$ returns z to sim_A.

sim_A chooses z_0', z_1, $\cdots$, z_l uniformly at random, and then computes the corresponding commitment for z_0' and z_i $(1 \leq i \leq l)$.

For the fixed string f, sim_A then chooses $\tilde{z}$ uniformly at random, and then computes z_0 from the equation

$$\tilde{z} + fz = (z_0, z_1, \cdots, z_l) \cdot (1, x_1, \cdots, x_l)$$

Notice that the commitment of each z_i is correct except for the commitment of z_0'. Thus, sim_A must compute a valid commitment of z_0 from the equation

$$g_1^{z_0'} g_2^{r_{z_0'}} = g_1^{z_0} g_2^{r_{z_0}}$$

This is possible since the simulator sim_A knows the auxiliary information of the commitment scheme. It follows that for any polynomial time distinguisher D, the view of D when it interacts with A in real conversation is computationally indistinguishable from that when it interacts with sim_A in the ideal world.

Lemma 2. *For any malicious Bob B, there exists a simulator sim_B that plays the role of B in the ideal world such that for any probabilistic polynomial time distinguisher D, the view of D when it interacts with B in real conversation is computationally indistinguishable from that when it interacts with sim_B in the ideal world.*

Proof: sim_B rewinds Bob to extract the input vector $(y_0, y_1, \cdots, y_l)$ and the associated commitment vector $(\tilde{y_0}, \tilde{y_1}, \cdots, \tilde{y_l})$.

sim_B then gives the input vector $inp_B = (y_0, y_1, \cdots, y_l)$ to $\mathcal{F}_{OSP}$.

$\mathcal{F}_{OSP}$ returns z to sim_B.

Notice that in the following steps, Bob learns no new information. The simulator can just play the game following Alice's part of the protocol. The remaining question is whether the protocol ensures $z = \sum_{i=0}^{l} x_i y_i$. By rewinding Bob, sim_B obtains two commitment vectors $(z_0, z_1, \cdots, z_l)$ and $(z_0', z_1', \cdots, z_l')$ such that

$$\tilde{z} + fz = (z_0, z_1, \cdots, z_l) \cdot (1, x_1, \cdots, x_l)$$

$$\tilde{z} + f'z = (z_0', z_1', \cdots, z_l') \cdot (1, x_1, \cdots, x_l)$$

Consequently, we have the following equation by applying the binding property of the underlying commitment scheme

$$(f' - f)z = (z_0' - z_0) + (z_1' - z_1)x_1 + \cdots + (z_l' - z_l)x_l$$

i.e.,

$$(f' - f)z = (f' - f)(y_0 + y_1 x_1 + \cdots + y_l x_l)$$

It follows that

$$z = y_0 + x_1 y_1 + \cdots + x_l y_l$$

Combining Lemma 1 and Lemma 2, we have the main statement below:

Theorem 1. *Our implementation described above is secure against static adversary who corrupts Alice or Bob in the common reference string model assuming that the underlying Fujisaki-Okamoto's commitment scheme is statistically hiding and computationally binding, and Paillier's encryption scheme is semantically secure.*

4 Conclusion

In this paper, we have introduced and formalized a new notion called oblivious scalar-product protocols. We also have proposed an efficient implementation of oblivious scalar-product evaluation protocols that does not emulate the circuit for the functions and shown that our implementation is provably secure assuming that the underlying Fujisaki-Okamoto's commitment scheme is statistically hiding and computationally binding, and Paillier's encryption scheme is semantically secure in the common reference string model. Finally, several immediate applications of this new primitive have been proposed.

References

1. M.Abadi, J.Feigenbaum. Secure circuit evaluation: A protocol based on hiding information from an oracle. Journal of Cryptology 2 (1990), 1-12.
2. F.Boudot: Efficient Proofs that a Committed Number Lies in an Interval. Proc. of EUROCRYPT 2000: 431-444, Springer Verlag.
3. M.Ben-or, S.Golawasser, A.Wigderson. Completeness theorems for non-cryptographic faulttolerant distributed computation. In Proc. 20th Annual A CM Symposium on Theory off Computing (STOC) (1988), pp. 1-10.
4. R.Canetti: Universally Composable Security: A New Paradigm for Cryptographic Protocols. FOCS 2001: 136-145.
5. R.Cramer, I.Damgård: Secret-Key Zero-Knowlegde and Non-interactive Verifiable Exponentiation. TCC 2004: 223 - 237.
6. D.Chaum, C.Crépeau, and I.Damgård. Multiparty unconditionally secure protocols. In Proc. 20th Annual ACM Symposium on Theory off Computing (STOC) (1988), pp. 11-19.
7. I.Damgård, E.Fujisaki: A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. Proc. of ASIACRYPT 2002: 125-142, Springer Verlag.
8. W.Du, Z.Zhan, Building decision tree classifier on private data, In Proceedings of the IEEE ICDM Workshop on Privacy, Security and Data Mining (2002).
9. E.Fujisaki, T.Okamoto. Statistically zero knowledge protocols to prove modular polynomial relations. Crypto'97. 16-30, 1997.

10. O.Goldreich, S.Micali, A.Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Proc. 19th Annual A CM Symposium on Theory of Computing (STOC) (1987), pp. 218-229.
11. B.Goethals, S.Laur, H.Lipmaa, T.Mielikäinen: On Private Scalar Product Computation for Privacy-Preserving Data Mining. ICISC 2004: 104-120.
12. A.Kiayias, A. Mitrofanova: Testing Disjointness of Private Datasets. Financial Cryptography 2005: 109-124
13. M.Naor and B.Pinkas: Oblivious Transfer and Polynomial Evaluation. STOC 1999: 245-254.
14. P.Paillier: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Proc. of EUROCRYPT 1999: 223-238, Springer Verlag.
15. J.Vaidya, C.Clifton, Privacy preserving association rule mining in vertically partitioned data, In the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002), 639 -644.
16. A.C.Yao: How to Generate and Exchange Secrets (Extended Abstract) FOCS 1986: 162 - 167.
17. H.Zhu, F.Bao: Augmented Oblivious Polynomial Evaluation Protocol and Its Applications. ESORICS'05: 222 -230.
18. H.Zhu, F.Bao, T.Li, Y.Qiu: More on Shared-Scalar-Product Protocols. ISPEC'06: 142 -152.

On Optimizing the *k*-Ward Micro-aggregation Technique for Secure Statistical Databases

Ebaa Fayyoumi[1] and B. John Oommen[2]

[1] School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6
efayyoum@scs.carleton.ca
[2] *Professor* and *Fellow of the IEEE*, School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6
oommen@scs.carleton.ca

Abstract. We consider the problem of securing a statistical database by utilizing the well-known micro-aggregation strategy, and in particular, the *k*-Ward strategy introduced in [1] and utilized in [2]. The latter scheme, which represents the state-of-the-art, coalesces the sorted data attribute values into groups, and on being queried, reports the means of the corresponding groups. We demonstrate that such a scheme can be optimized on two fronts. First of all, we minimize the computations done in evaluating the between-class *distance* matrix, to require only a constant number of updating distance computations. Secondly, and more importantly, we propose that the data set be partitioned recursively before a *k*-Ward strategy is invoked, and that the latter be invoked on the "primitive" sub-groups which terminate the recursion. Our experimental results, done on two benchmark data sets, demonstrate a marked improvement. While the information loss is comparable to the *k*-Ward micro-aggregation technique proposed by Domingo-Ferrer[1] *et.al.* [2], the computations required to achieve this loss is a fraction of the computations required in the latter - providing a computational advantage which sometimes exceeds 80% if one method is used by itself, and more than 90% if both enhancements are invoked simultaneously.

1 Introduction

A lot of attention has been recently dedicated to the problem of maintaining the confidentiality of statistical databases through the application of statistical tools, so as to limit the identification of information on individuals and enterprises. The objective in statistical databases is to guarantee the confidentiality of the information provided, and also to provide useful statistical summaries of the data to the user. Thus, *Disclosure Risk* and *Information Loss* are the most important criteria for secure statistical databases. Minimizing the disclosure risk would result in minimizing the risk to the confidentiality of the entities involved in the released data. Additionally, the utility of the data should be maximized, which implies increasing the value and validity of the released data to a legitimate data

[1] The authors are extremely grateful to Dr. Domingo-Ferrer for his assistance.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 324–335, 2006.
© Springer-Verlag Berlin Heidelberg 2006

user. The goal of the exercise is to optimize these potentially conflicting criteria by reaching an equilibrium point between them. Several methods, presented in [3], have been proposed for limiting the disclosure risk.

One of the recent techniques involves the strategy called "Micro-aggregation". The latter comprises a family of statistical disclosure limitation techniques used to protect micro-data files containing records on individual data subjects. These belong to the family of substitution/perturbation approaches [2, 4, 5, 6], where individual values are replaced by values computed on small aggregates *prior* to publication. The confidentiality of the individual data subjects is protected by ensuring that each group has at least a minimum number of observations, k. The objective of this technique is to group similar records together, so that the replacement of actual values by the means of *their* associated groups will result in minimizing the information loss [7, 8].

The micro-aggregation problem as formulated in [2, 6, 7, 8], can be stated as follows: A micro-data set $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ is specified in terms of the n "individuals", namely the $U_i's$, each representing a data vector whose components are p continuous variables. Each data vector can be viewed as $U_i = [u_{i1}, u_{i2}, \ldots, u_{ip}]^T$, where u_{ij} specify the value of the j^{th} variable in the i^{th} data vector. Associated with the problem is a positive integer, k, referred to as the *security parameter*. Micro-aggregation involves partitioning the n data vectors into m groups so as to obtain a k-partition $\mathbb{P}_k = \{G_i \mid 1 \leq i \leq m\}$, such that each group, G_i, of size n_i, contains between k and $2k-1$ data vectors, and each data vector is contained in exactly one group, implying that $\mathbb{U} = \bigcup_{i=1}^{m} G_i$ and $G_i \bigcap G_j = \emptyset$. The j^{th} data vector in the i^{th} group is denoted by X_{ij} (where each X_{ij} is an element of $\mathcal{U}$), while $\bar{X}_i$ is the average of the data vectors over the i^{th} group, and $\bar{X}$ is the average of the data vectors over the entire set of all the n elements of $\mathcal{U}$. Thus, $\bar{X}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} X_{ij}$, and $\bar{X} = \frac{1}{m} \sum_{i=1}^{m} \bar{X}_i$.

The optimal k-partition $\mathbb{P}_k^*$ is defined to be the one that maximizes the within-group homogeneity, since a large value of this index implies a smaller information loss. The within-group similarity is defined as the *Sum of Squares Error* (SSE) computed on the basis of the Euclidean distances of each individual data vector X_{ij} to the centroid $\bar{X}_i$ of the group to which it belongs, and is given by: $SSE = \sum_{i=1}^{m} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^T (X_{ij} - \bar{X}_i)$. Analogously, the between-groups similarity is defined as the *Sum of Squares Among* the groups (SSA), which is the squared deviations of the means from the total mean, and is given as: $SSA = \sum_{i=1}^{m} n_i (\bar{X}_i - \bar{X})^T (\bar{X}_i - \bar{X})$. The *Total Sum of Squares* is denoted by $SST = SSA + SSE$. The *Information Loss* is quantified as: $L = \frac{SSE}{SST}$.

1.1 Contribution of the Paper

The main contribution of this paper is to demonstrate that the speed of data Micro-Aggregation Techniques ($MATs$) can be increased by two considerations, namely that of invoking recursive computations (which is crucial in large-sized data sets), and by utilizing only the values that lie on the diagonal above the main diagonal of the so-called distance matrix. This, in turn, is achieved by two enhancements to the k-Ward algorithm: namely Recursive k-Ward (kWR) and

k-Ward Diagonal (kWD), respectively. This paper demonstrates the power of these modifications in increasing the speed of the strategy while preserving the information loss.

It should be mentioned that by using our philosophy, a recursive method can actually be applied to any known MAT. We believe that the power of such a method will be clearly evident for the multivariate data sets, and more specifically in the implementation of the Multivariate Fixed Size method (MFS), and the Multivariate Data-Oriented micro-aggregation (MDO) that invoke a k-Ward's philosophy [9], and that the gleaned advantage will be marked especially for "large" data sets. Since the distance matrix involving the set of data vectors cannot be computed and stored *a priori*[2], these distances and the resulting matrix must be computed "on demand"- i.e., as and when they are needed. This, of course, demands a considerable computational overhead [9], which is significantly reduced by using the optimized version of the k-Ward's method.

The combination of invoking a recursive computation and requiring a small subset of the distance computation are novel- we are not aware of any comparable results. However, the remarkable reduction in computation (sometimes as much as 80% if one method is used by itself, and more than 90% if both enhancements are invoked simultaneously) renders the contribution of the paper significant.

2 The State-of-the-Art

The review of the state of art is fairly limited, and includes only those pieces of literature that directly pertain to our work. A formal algorithm to find the optimal solution for the k-partition problem which minimized the information loss, L, was proposed by Defays and Nanopoulos in [11].

Practical heuristic-based MATs have been proposed in [11, 12]. The partitioning mechanism advocated (for uni-dimensional data) in all these papers is the same: First of all, the elements are ranked in an ascending or descending order, and subsequently, groups of k consecutive values assumed by the variable in question are replaced by their average. If the total number of elements n is not a multiple of k, the last group or the group containing the modal value will contain more than k elements [2].

Initial research in the field proposed "fixed " $MATs$ which required that the size of each partition group was a fixed constant, k. These, in turn, led to the *Fixed-Size Micro-aggregation* algorithms [11, 12]. Recent developments [2, 9] have concentrated on further reducing the information loss by using variable-sized data-dependent groups, leading to families of *Data-Oriented Micro-aggregation* algorithms. The philosophy that is utilized underlies the fact that groups need not consist of exactly k data vectors, but of *at least* k data vectors. They also preserve the natural data aggregate by allowing the group size to be between

[2] Agusti Solanas *et.al.* have recently investigated that the distance matrix can be stored when the number of records is very large by applying the blocking technique [10].

k and $2k-1$, depending on the structure of the data, so as to lead to more homogeneous groups and to, thus, minimize the information loss [2, 5, 6, 13].

Two alternative heuristic approaches which incorporate variable-size micro-aggregation have been presented in [2, 9, 15]. The authors of [2] presented a genetic algorithm that appears as an alternative heuristic to offer a linear complexity. It presents the k-partitions as a binary string (the "chromosomes"), and combines directed and random search strategies to attempt to attain a global optimum. A hierarchical classification method can be used to obtain building blocks for the heuristic $MATs$ that yield variable-sized groups. Ward's method, proposed in [1], is attractive because it is stepwise optimal: Two groups or data elements coalesced at each step are chosen so that the increase in the within-groups index, the SSE, caused by their union is minimal. However, Ward's method had to be adapted into the so-called k-Ward's method to render it applicable for micro-aggregation [9, 15]. Then the k-Ward scheme was extended to a *Secure-k Ward* [14] in order to enhance the individual's privacy. An efficient polynomial algorithm which uses graph techniques to solve the univariate micro-aggregation problem can be found in [7].

3 *k*-Ward Micro-Aggregation Technique

Ward's agglomerative hierarchical clustering method described above [1, 17] has been modified to provide an optimal k-partition solution by enforcing the group-size constraint [2, 9, 15]. Such a k-Ward_MAT can be applicable for quantitative data, and for qualitative data when an appropriate distance is defined. The authors of [2] proved that the k-Ward's algorithm terminates after a finite number of steps, and that the computational complexity is quadratic (i.e., $\mathrm{O}(n^2)$).

Based on the Ward algorithm, the authors of [2, 9] have developed the k-Ward_MAT algorithm for micro-aggregation. The essential qualifiers for this are the facts that the distance criterion is the SSE, and that each group should contain between k and $2k-1$ elements. The k-Ward_MAT algorithm is included in [2, 6, 9, 16]. The experimental results related to k-Ward_MAT are found in [2], and also included in Section 5 of this paper.

4 Optimized *k*-Ward Micro-Aggregation Technique

The authors of [2] claim that the performance of MATs depends on the distribution of the variables. However, it has been observed that no single method outperforms all other methods for all variables involving real life data. This conclusion was made after testing all the competing methods for each variable, and using the method which yielded the lowest information loss. In this paper, we restrict our study to the k-Ward scheme, because it demonstrates a good trade-off between the information loss and the data disclosure. In addition to this, it also possesses the attractive feature that it permits a natural extension to multivariate micro-aggregation. In this regard, we observe that there is much room for improvement when it concerns the speed of micro-aggregation. Indeed,

the current k-Ward's method suffers from two major disadvantages, firstly, that of the excessive computational burden encountered by processing *all* the data elements, and additionally that of computing the "entire" distance matrix which contains the distances between every single pair of groups. Therefore, we propose two modifications by which we can optimize this method, and reduce the required time needed to micro-aggregate the data set.

4.1 Optimizing Distance-Based Computations

The first modification we achieve is in the Ward's method itself [1], and more specifically, in the phase which computes the distance matrix. The latter is an $m \times m$ matrix, where m represents the number of groups included at each step. It contains the distance values between the groups (recorded as D_{ij}) which, in turn, represents the SSE obtained by potentially merging G_i with G_j. This matrix is symmetric, and has a zero diagonal.

The k-Ward's method requires a number of basic steps, in order to generate a (near) optimal k-partition, where the best-case number of steps is $n(1-1/k)$, and the worst-case number is $(n/k-1)(n/2+k-2)$ [2]. At each basic step, the number of groups is reduced by unity. Currently, there are two different approaches to compute the value of the distance matrix, using either a stored matrix approach, or by invoking a stored data approach. In the k-Ward's method, which uses a stored matrix approach[3], $(m^2-m)/2$ values are computed at the initialization step, and $m-1$ values are recomputed at each basic step. But using our newly introduced enhanced version of k-Ward_MAT, the so-called k-Ward Diagonal, (kWD), only $m-1$ values are computed in the initial step, and at most 2 values are recomputed during a basic step. On the other hand, using a stored data approach[4] for a k-Ward scheme, requires no initialization step, and it computes $(m^2-m)/2$ values at each basic step. As opposed to this, kWD computes only $m-1$ values which lie on the diagonal above the main zero diagonal.

In principle, kWD behaves just like k-Ward_MAT, but the primary difference involves the way Ward's algorithm is invoked. While k-Ward_MAT invokes the Ward algorithm, kWD avoids computing *all* the values above the main diagonal in order to find the nearest pair of distinct groups. Rather, it computes only the SSE values that lie on the principal diagonal above the main zero diagonal. This is clearly included in Step 2 of the algorithm shown below.

The rationale for optimizing the computation as in the kWD is based on the mathematical results given presently.

Since we are grouping an ordinal data set, we present a result that states that the distance value of merging G_i with G_{i+j} is less than the distance value of merging G_i with G_{i+k}, when $k > j$ with a very high probability. This is shown in Theorem 1 which needs the result of Lemma 1 below. The proofs are included in [16] and omitted here in the interest of brevity.

[3] The distance matrix is computed, stored, and retrieved from storage as needed.
[4] The distance matrix is computed when needed rather than retrieved from storage.

Algorithm 1. kWD

Input: A set of *sorted* data records
Output: The groups of micro-aggregated records
Method:

1: Form a group from the first (smallest) k elements of the data set, and another group with the last (largest) k elements of the data set. Initialize the intermediate elements so as to constitute single-element groups.
2: Use Ward's method until all elements in the data set belong to a group containing k or more data elements. In the process of forming groups by Ward's method, the criterion used is the SSE. However, rather than computing all the values above the main diagonal of the inter-group distance matrix, we compute only the values that lie on the diagonal above the main diagonal. Also, in the process, never merge two groups *both* of which have a size greater than or equal to k.
3: **for** each group in the final partition that contains $2k$ or more data elements **do**
4: Apply this algorithm again to particular group containing $2k$ or more elements.
5: **end for**
6: **return** the set of groups and report the mean of the group (on being queried).
7: **End Algorithm** kWD

Lemma 1. *Let* $\mathbb{P}_k^* = \{G_1, G_2, G_3, \ldots, G_m\}$ *be an optimal partition. Then each partition* $G_i = \{X_{i1}, X_{i2}, \ldots, X_{in_i}\}$ *satisfies this property* $X_{in_i} \leq X_{j1}$, *where* $i < j$.

Theorem 1. *Consider the quantity* $D(G_i, G_j)$, *defined as* $\frac{n_i n_j}{n_i + n_j} (\bar{X}_i - \bar{X}_j)^2$. *Then, if the index* $k > j$,*and the size of the groups* G_{i+k} *and* G_{i+j} *satisfy* $n_{i+k} \leq n$ *and* $n_{i+j} \leq n$ *respectively,*

$$\begin{cases} D(G_i, G_{i+k}) \geq D(G_i, G_{i+j}), & \text{Whenever } n_{i+k} \geq n_{i+j} \\ D(G_i, G_{i+k}) \geq D(G_i, G_{i+j}), & \text{Whenever } n_{i+k} < n_{i+j} \text{ and if the groups} \\ & \quad G_{i+j} \text{ and } G_{i+k} \text{ have different elements} \\ D(G_i, G_{i+k}) < D(G_i, G_{i+j}), & \text{Otherwise.} \end{cases}$$

Theorem 1 can be used to reduce the computational time required for determining the nearest pair of distinct groups. The condition $D(G_i, G_{i+k}) \geq D(G_i, G_{i+j})$ is almost always true, except in a single case where both groups G_{i+j} and G_{i+k} have elements with a single repetitive entry - which is an event occurring with a very small probability. Thus, if this event is ignored, it is obvious that computing the distances between G_i and G_{i+1} for all $i \leq n-1$ is sufficient to determine the minimum distance value in the distance matrix. The values of these distances lie on the principal diagonal above the main diagonal due to the ordering of the elements. This is what the kWD algorithm achieves, thus significantly reducing the computational time.

4.2 Recursive k-Ward Optimization

Having discussed how the k-Ward computation can be optimized for a single partitioning, we now propose a recursive and superior mechanism, referred to as

Algorithm 2. kWR

Input: InSet: A set of sorted data set records; θ: The user-defined threshold; J: A fixed constant.
Output: OutSet: The micro-aggregated records
Method:

Partition $InSet$ into J mutually exclusive Sets $InSet_1, InSet_2, \ldots, InSet_J$
if $\theta > \frac{\Sigma_{i=1}^{J} SST(InSet_i)}{SST(InSet)}$ **then**
  call k-Ward ($InSet$)
  return $OutSet$
else
  for $i = 1 \leftarrow J$ **do**
    call Recursive-k-Ward_MAT ($InSet_i$,$OutSet_i$)
  end for
end if
return $OutSet = OutSet_1 \bigcup OutSet_2 \cdots \bigcup OutSet_J$
End Algorithm kWR

Recursive k-Ward Micro-aggregation (kWR) for further minimizing the computations for the entire data set. Our strategy is the following: Rather than process all the data using a k-Ward method, we propose that the data be recursively subdivided into smaller subsets. We emphasize that the smaller subsets need not be obtained as the result of invoking a k-Ward algorithm on the original data. This philosophy leads to a sequence of so-called divide-k-Ward-coalesce steps which are invoked recursively to ultimately yield the desired micro-aggregated records. Furthermore, we propose that each subset be micro-aggregated independently. Finally, the micro-aggregated records are combined in order to obtain the entire set of records appropriately grouped. The algorithm is presented below.

The reader should observe that this philosophy is *quite distinct* from the partitioning that uses *prior* clustering methods. Because the recursive method is able to examine the spread of the data set, when there is a large gap in the data, such a scheme will be able to force the gap to be *between* the groups rather than within the groups. This is achieved by dividing the whole data set into a number of disjoint subsets as dictated by the "gap" that exists.

Using a recursive strategy ensures that at least J subsets are obtained from the entire data set, which leads to a noticeable saving on time. Further, by this recursive method, we not only reduce the required time to micro-aggregate the records, but also attempt to optimally preserve the minimization of the information loss. This is achieved by invoking the base (terminating) step, where the information loss is minimized for each atomic partition.

5 Experimental Results

In this Section, we describe our data sets, explain our experimental methods, and present the results we have obtained by using kWD, kWR and $kWDR$ for micro-aggregation.

5.1 Data Sets

We tested the various versions of the k-Ward_MAT using two real data sets that have been used as benchmarks in previous studies [2, 18]: (i)**Tarragona Data Set** [2], which involves 834 companies in the Tarragona area. Each company, has 13 associated quantitative variables referred to as $Var1$ to $Var13$. (ii)**Census Data Set** [18], which was obtained on $July$ 27, 2000 using the Data Extraction System of the U.S. Bureau of the Census. It contains 1,080 records obtained as per the extraction procedure described in [19], and has 13 quantitative variables referred to as $Var1$ to $Var13$.

5.2 Results

The run time characteristics of the Optimized k-Ward micro-aggregation algorithms for both the real-life data sets are discussed in more detail in this Section. We set the value of k =3 in all the experiments.

The sorted data approach was chosen to obtain the specific implementation of kWD. Table 1 presents the result of executing kWD. From the Table we see that kWD leads to a huge reduction in the time required to micro-aggregate all the records in the data set compared to the original k-Ward_MAT. For example, the required time to micro-aggregate the values of $Var11$ in the Tarragona data set using k-Ward_MAT was 19.86 seconds while it was only 3.95 seconds using kWD. The percentage of time improvement reaches up to 80.11% on the Tarragona data set. Similarly, the required time to micro-aggregate all values of $Var2$ in the Census data set was 42.41 seconds using k-Ward_MAT, but the kWD required only 8.36 seconds. Again, the percentage of time improvement is as high as 80.29% on the Census data set. Interestingly, both the k-Ward_MAT and kWD obtain the same value of the information loss for each variable in both the data sets.

In the case of kWR, the data set was partitioned into 2 subsets ($J = 2$) whenever the base condition was not satisfied. It should be clarified that finding the best threshold value, θ, is far from trivial. Indeed, the value of θ plays an important role in determining the required computational time, and also in determining the value of the information loss. Thus, if θ has a certain value (that does not allow the base condition to be satisfied) it is clear that the recursion will be invoked more often, leading to a huge reduction in the running time, but to an increased value in the information loss. It is important to mention that the number of recursions invoked differs from one variable to another, but for each specific variable, the number of recursive calls seems to be the same regardless of the value of k. This makes sense because the recursion divides the data set into subsets as per the base condition before invoking the micro-aggregation method. Besides, we also observe that the base terminating condition is generally satisfied at the same positions for each specific variable regardless of the value of k. Finally, we have observed that changing the value of k effects the value of the information loss and the running time, but does not seem to effect the total number of recursive invocations.

Tables 2 shows a comparison between the original k-Ward_MAT and kWR on the Tarragona data set. In the Table, the sequence of recursive calls is repre-

Table 1. Comparison between the original k-Ward_MAT and the optimized kWD on the Tarragona and Census data sets

Tarragona Data Set						Census Data Set					
Var	kW		kWD			Var	kW		kWD		
	Info. Loss	Time	Info. Loss	Time	Improv. (%)		Info. Loss	Time	Info. Loss	Time	Improv. (%)
Var1	7.176	19.89	7.176	5.41	72.80%	Var1	0.131	42.44	0.131	9.48	77.66%
Var2	0.557	19.72	0.557	4.98	74.75%	Var2	0.001	42.41	0.001	8.36	80.29%
Var3	0.775	19.77	0.775	4.99	74.76%	Var3	0.008	42.58	0.008	8.42	80.23%
Var4	1.487	19.61	1.487	4.88	75.11%	Var4	0.005	42.03	0.005	8.84	78.97%
Var5	2.018	49.03	2.018	12.78	73.93%	Var5	0.024	41.92	0.024	8.91	78.75%
Var6	0.586	19.69	0.586	5.03	74.45%	Var6	0.034	42.00	0.034	9.14	78.24%
Var7	1.921	19.81	1.921	5.03	74.61%	Var7	0.002	42.06	0.002	8.36	80.12%
Var8	0.338	19.73	0.338	4.95	74.91%	Var8	0.443	41.69	0.443	10.66	74.43%
Var9	1.287	20.02	1.287	5.47	72.68%	Var9	0.732	44.84	0.732	10.05	77.59%
Var10	1.905	19.69	1.905	4.67	76.28%	Var10	0.003	47.02	0.003	10.34	78.01%
Var11	2.966	19.86	2.966	3.95	80.11%	Var11	0.008	47.11	0.008	10.28	78.18%
Var12	4.748	19.72	4.748	4.72	76.06%	Var12	0.004	47.31	0.004	10.28	78.27%
Var13	4.958	19.84	4.958	4.83	75.66%	Var13	0.005	48.34	0.005	10.59	78.09%

sented by an expression of well-matched parentheses. Thus, for example, there are two recursive calls for the variable $Var2$ in the Tarragona data set, and this is represented by the parenthetic string "(()(()()))". The first time the algorithm is invoked, it divided the entire data set into two partitions, each recursively invoking the algorithm again. The first partition was continued with no further divisions or recursions, while the second partition was divided into another two sub-partitions and the algorithm was recursively invoked for each sub-partitions, which ultimately terminated the process. It turns out that in this case the value of $\theta = 0.8$ is suitable for all the 13 variables of the Tarragona data set, and the percentage of the time improvement reaches up to 93.65% on $var12$ after 8 recursive calls. But for the Census data set, we observe that we have to set θ differently for each variable, and that values of θ which are respectively 0.4, 0.28, 0.31, 0.28, 0.31, 0.38, 0.29, 0.70, 0.80, 0.31, 0.27, 0.32, and 0.30 (respective to the corresponding variables $Var1$ to $Var13$) lead to the best solution. kWR yielded the same value of the information loss in most of these variables except for the cases of $var4$, $Var5$, $Var6$, $Var11$, and $Var12$ where it gave a value close to the value of information loss obtained by the original k-Ward_MAT. We believe that the reason for this minor deviation from the optimal is due to the nature of the data. An examination of the values of data in the Tarragona data set shows that since they are relatively close to each other, there are not many gaps between the values. But in the case of the Census data set such gaps exist. Thus, there are many divisions on the Census data set, which will marginally increase the information loss. However, we still believe that such a small loss is worth the significant computation gain. Additional results concerning the Census data set are found in [16].

Table 2. Comparison between the original k-Ward_MAT and the optimized kWR using both a fixed threshold and dynamic threshold on the Tarragona data set

Var	kW		kWR - Fixed threshold				kWR - Dynamic threshold			
	Info. Loss	Time	Info. Loss	Time	Improv (%)	Recursion	Info. Loss	Time	Improv (%)	Recursion
Var1	7.176	19.89	7.176	5.31	73.30%	((()(()(())))	7.176	10.30	48.22%	((()))
Var2	0.557	19.72	0.557	3.75	80.98%	((()(())))	0.557	5.78	70.69%	((()))
Var3	0.775	19.77	0.775	5.91	70.11%	((()))	0.775	5.78	70.76%	((()))
Var4	1.487	19.61	1.487	1.49	92.40%	(((((((())))))))(()()))	1.487	5.66	71.14%	((()))
Var5	2.018	49.03	2.018	22.36	54.40%	((()(())))	2.018	24.89	49.24%	((()))
Var6	0.586	19.69	0.586	3.70	81.21%	((()(())))	0.586	5.70	71.05%	((()))
Var7	1.921	19.81	1.922	3.75	81.07%	((()(())))	1.921	5.78	70.82%	((()))
Var8	0.338	19.73	0.338	3.75	80.99%	((()(())))	0.338	5.72	71.01%	((()))
Var9	1.287	20.02	1.287	3.83	80.87%	((()(())))	1.287	5.91	70.48%	((()))
Var10	1.905	19.69	1.909	1.23	93.75%	(((((()))))))(()())))	1.905	5.76	70.75%	((()))
Var11	2.966	19.86	2.967	1.99	89.98%	(((()))(())))	2.966	1.78	91.04%	((((()))(()))))
Var12	4.748	19.72	4.748	1.19	93.97%	((((((((()))))))))(()())))	4.748	3.88	80.32%	(((())))
Var13	4.958	19.84	4.958	1.26	93.65%	((((((((()))))))))(()())))	4.958	3.73	81.20%	(((())))

Table 3. Comparison between the original k-Ward_MAT and the optimized $kWDR$ which is a simultaneous combination of both kWD and kWR on the Tarragona (using a fixed threshold) and Census (using a dynamic threshold) data sets

Tarragona Data Set						Census Data Set					
Var	kW		$kWDR$			Var	kW		$kWDR$		
	Info. Loss	Time	Info. Loss	Time	Improv. (%)		Info. Loss	Time	Info. Loss	Time	Improv. (%)
Var1	7.176	19.89	7.176	3.08	84.51%	Var1	0.131	42.44	0.131	13.39	68.45%
Var2	0.557	19.72	0.557	1.23	93.76%	Var2	0.001	42.41	0.001	2.42	94.29%
Var3	0.775	19.77	0.775	2.16	89.07%	Var3	0.008	42.58	0.008	2.03	95.23%
Var4	1.487	19.61	1.487	0.59	96.99%	Var4	0.005	42.03	0.005	2.20	94.77%
Var5	2.018	49.03	2.018	5.70	88.37%	Var5	0.024	41.92	0.024	2.52	93.99%
Var6	0.586	19.69	0.586	1.22	93.80%	Var6	0.034	42.00	0.034	2.27	94.60%
Var7	1.921	19.81	1.921	1.22	93.84%	Var7	0.002	42.06	0.002	4.28	89.82%
Var8	0.338	19.73	0.338	1.24	93.72%	Var8	0.443	41.69	0.443	4.39	89.47%
Var9	1.287	20.02	1.287	1.31	93.46%	Var9	0.732	44.84	0.732	5.67	87.36%
Var10	1.905	19.69	1.909	0.56	97.16%	Var10	0.003	47.02	0.005	5.70	87.88%
Var11	2.966	19.86	2.967	0.81	95.92%	Var11	0.008	47.11	0.011	4.59	90.26%
Var12	4.748	19.72	4.748	0.52	97.36%	Var12	0.004	47.31	0.008	6.55	86.16%
Var13	4.958	19.84	4.958	0.56	97.18%	Var13	0.005	48.34	0.009	6.92	85.68%

5.3 Dynamic Threshold

Since we are attempting to reduce the computational time and the information loss, we have also experimented the schemes using a dynamic threshold, which varies at each step. This threshold was computed as: $\theta = \frac{SST(InSet_1)+SST(InSet_2)}{SST(InSet)}$.

At each step, the dynamic threshold was passed as a *parameter* to the invoked recursion. Table 2 shows the power of the dynamic threshold scheme which yields

the same value of the information loss obtained by original k-Ward_MAT on the Tarragona data set. The results concerning the Census data set are found in [16]. The computation time is also reduced significantly, and thus, for example, the percentage of the time improvement reaches up to 91.04% on Var 11 in the Tarragona data set, while it is 45.44% on Var 3 in the Census data set.

Since it is prudent to investigate the computational advantage obtained by using a combination of the two modifications, kWD and kWR, we have also devised an augmented modification, $kWDR$, which is the overall optimized version of the original k-Ward_MAT. Thus, $kWDR$ computes the micro-aggregation by recursive calls, and in each case computes only the distance values advocated by Theorem 1. The results of $kWDR$ are shown in Table 3. The reduction of time reaches up to 97.36% on the Tarragona data set by using a combination of kWD and kWR with a fixed threshold, while a combination of kWD and kWR with a dynamic threshold yields 95.23% advantage on the Census data set.

6 Conclusions

In this paper, we have considered the problem of securing a statistical database. We have resorted to the the well-known micro-aggregation philosophy, and in particular, considered the k-Ward strategy, which coalesces the sorted data attribute values into groups, and on being queried, reports the means of the corresponding groups. We have shown that such a scheme can be optimized by minimizing the computations done in evaluating the between-class *distance* matrix, and by recursively partitioning the data set before k-Ward strategy is invoked, and that the latter be invoked on the "primitive" sub-groups which terminate the recursion. Our experimental results, done on two benchmark data sets, report a marked improvement. While the information loss is comparable to the k-Ward_MAT proposed by Domingo-Ferrer *et.al.* [2], the computations required to achieve this loss is a fraction of the computations required in the latter - providing a computational advantage which sometimes exceeds 80% if one method is used by itself, and more than 90% if both enhancements are invoked simultaneously.

References

1. Ward, J.H.: Hierarchical grouping to optimize an objective function. J. American Statistical Association **58** (1963) 236–245
2. Domingo-Ferrer, J., Mateo-Sanz, J.M.: Practical data-oriented microaggregation for statistical disclosure control. IEEE Transactions on Knowledge and Data Engineering **14** (2002) 189–201
3. Adam, N.R., Wortmann, J.C.: Security-control methods for statistical databases: A comparative study. ACM Computing Surveys **21** (1989) 515–556
4. Baeyens, Y., Defays, D.: Estimation of variance loss following microaggregation by the individual ranking method. In: Proceedings of Statistical Data Protection'98, Luxembourg: Office for Official Publications of the Eur. Comm. (1999) 101–108

5. Cuppen, M.: Sourec Data Perturbation in Statistical Disclosure Control. PhD thesis, Statistics Netherlands (2000)
6. Mateo-Sanz, J.M., Domingo-Ferrer, J.: A method for data-oriented multivariate microaggregation. In: Proceedings of Statistical Data Protection'98, Luxembourg: Office for Official Publications of the European Communities (1999) 89–99
7. Hansen, S.L., Mukherjee, S.: A polynomial algorithm for univariate optimal microaggregation. IEEE Trans. on Know. and Data Eng. **15** (2003) 1043–1044
8. Laszlo, M., Mukherjee, S.: Minimum spanning tree partitioning algorithm for microaggregation. IEEE Trans. on Know. and Data Eng. **17** (2005) 902–911
9. Mateo-Sanz, J.M., Domingo-Ferrer, J.: A comparative study of microaggregation methods. Questiio **22** (1998) 511–526
10. Solanas, A., Martínez-Ballesté, A., Domingo-Ferrer, J., Mateo-Sanz, J.: A 2d-tree-based blocking method for microaggregating very large data sets. In: The First International Conference on Availability, Reliability and Security. (2006)
11. Defays, D., Nanopoulos, P.: Panels of enterprises and confidentiality: the small aggregates method. In: Proceedings of 92 Symposium on Design and Analysis of Longitudinal Surveys, Ottawa: Statistics Canada (1993) 195–204
12. Defays, D., Anwar, N.: Micro-aggregation: A generic method. In: Proceedings of the 2nd International Symposium on Statistical Confidentiality, Luxembourg: Office for Official Publications of the European Communities (1995) 69–78
13. Solanas, A., Martínez-Ballesté, A.: V-mdav: A multivariate microaggregation with variable group size. In: 17th COMPSTAT Symposium of the IASC, Rome (2006)
14. Li, Y., Zhu, S., Wang, L., Jajodia, S.: A privacy-enhanced microaggregation method. In: FoIKS '02: Proceedings of the Second International Symposium on Foundations of Info. and Know. Sys., London, UK, Springer-Verlag (2002) 148–159
15. Domingo-Ferrer, J., Mateo-Sanz, J.M.: Resampling for statistical confidentiality in contingency tables. Comp. and Math. with App. **38** (1999) 13–32
16. Fayyoumi, E., Oommen, B.J.: (Enhancing k-ward micro-aggregation for secure statistical databases using distance-based and recursive optimizations) Unabridged Version of This Paper.
17. Brucker, P.: On the complexity of clustering problems. In Hehn, R., Korte, B., Oettli, W., eds.: Optimization and Operations Research (1977) 45–54
18. Domingo-Ferrer, J., Torra, V.: A quantitative comparison of disclosure control methods for microdata. In Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies, Berlin: Springer-Verlag (2002) 113–134
19. Brand, R., Domingo-Ferrer, J., Mateo-Sanz, J.M.: Reference data sets to test and compare SDC methods for protection of numerical microdata. Technical report, CASC PROJECT, Computational Aspects of Statistical Confidentiality (2002)

Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation Without Random Oracles

Eike Kiltz[1] and David Galindo[2]

[1] CWI Amsterdam
The Netherlands
kiltz@cwi.nl
[2] Radboud University Nijmegen
The Netherlands
d.galindo@cs.ru.nl

Abstract. We describe a new and practical identity-based key encapsulation mechanism that is secure in the standard model against chosen-ciphertext (CCA2) attacks. Since our construction is direct and not based on generic transformations from hierarchical identity-based encryption, it is more efficient than all previously proposed schemes.

1 Introduction

Identity-Based Encryption and Key Encapsulation. An Identity-Based Encryption (IBE) scheme is a public-key encryption scheme where any string is a valid public key. In particular, email addresses and dates can be public keys. The ability to use identities as public keys avoids the need to distribute public key certificates.

Instead of providing the full functionality of an IBE scheme, in many applications it is sufficient to let sender and receiver agree on a common random session key. This can be accomplished with an *identity-based key encapsulation mechanism* (IB-KEM) as formalized in [5]. Any IB-KEM can be updated to a full IBE scheme by adding a symmetric encryption scheme with appropriate security properties.

After Shamir proposed the concept of IBE in 1984 [27] it remained an open problem for almost two decades to come up with a satisfying construction for it. In 2001, Boneh and Franklin [8] proposed formal security notions for IBE systems and designed a fully functional secure IBE scheme using bilinear maps. This scheme and the tools developed in its design have been successfully applied in numerous cryptographic settings, transcending by far the identity based cryptography framework. IBE is currently in the process of getting standardized — from February 2006 on the new IEEE P1363.3 standard for "Identity-Based Cryptographic Techniques using Pairings" [19] accepts submissions.

An alternative but less efficient IBE construction was proposed by Cocks [14] based on quadratic residues. Both IBE schemes (Cocks' scheme only through Fujisaki-Okamoto [15]) provide security against *chosen-ciphertext attacks*. In

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 336–347, 2006.
© Springer-Verlag Berlin Heidelberg 2006

a chosen ciphertext attack, the adversary is given access to a decryption oracle that allows him to obtain the decryptions of ciphertexts of his choosing. Intuitively, security in this setting means that an adversary obtains (effectively) no information about encrypted messages, provided the corresponding ciphertexts are never submitted to the decryption oracle. For different reasons, the notion of chosen-ciphertext security has emerged as the "right" notion of security for encryption schemes. We stress that, in general, chosen-ciphertext security is a much stronger security requirement than chosen-plaintext attacks [3, 16], where in the latter an attacker is not given access to the decryption oracle.

The drawback of the IBE scheme from Boneh-Franklin and Cocks is that security can only be guaranteed in the *random oracle* model [4], i.e. in an idealized world where all parties magically get black-box access to a truly random function. Unfortunately a proof in the random oracle model can only serve as a heuristic argument and has proved to possibly lead to insecure schemes when the random oracles are implemented in the standard model (see, e.g., [12]).

Waters' IBE. To fill this gap Waters [29] presents the first efficient Identity-Based Encryption scheme that is chosen-plaintext secure without random oracles. The proof of his scheme makes use of an algebraic method first used by Boneh and Boyen [6] and security of the scheme is based on the Bilinear Decisional Diffie-Hellman (BDDH) assumption. However, Waters' plain IBE scheme only guarantees chosen-plaintext security.

From 2-level Hierarchical IBE to chosen-ciphertext secure IBE. Hierarchical identity-based encryption (HIBE) [18, 17] is a generalization of IBE allowing for hierarchical delegation of decryption keys.

Recent results from Canetti, Halevi, and Katz, further improved upon by Boneh and Katz [10] show a generic and practical transformation from any chosen-plaintext secure 2-level HIBE scheme to a chosen-ciphertext secure IBE scheme. Since Waters' IBE scheme can naturally be extended to a 2-level HIBE this implies the first chosen-ciphertext secure IBE in the standard model. Key size, as well as the security reduction of the resulting scheme are comparable to the ones from Waters' IBE. However, the transformation involves some symmetric overhead to the ciphertext in form of a one-time signature or a MAC with their respective keys.

1.1 Our Contributions

Our two main contributions can be summarized as follows.

A direct chosen-ciphertext secure IB-KEM based on Waters' IBE. Our main idea is to enhance (the IB-KEM version of) Waters *chosen-plaintext* secure IBE by adding some redundant information to the ciphertext (consisting of a single group element) to make it *chosen-ciphertext* secure. This information is used to check whether a given IB-KEM ciphertext was "properly

generated" by the encryption algorithm or not; if so decryption is done as before, otherwise the ciphertext is simply rejected. Intuitively, this "consistency check" is what gives us the necessary leverage to deal with the stronger chosen-ciphertext attacks. Unfortunately implementing the consistency check is relatively expensive and an equivalent "implicit rejection" method is used to improve efficiency.

This provides the first direct construction of a chosen-ciphertext secure IB-KEM that is not explicitly derived from hierarchical techniques. No exogenous consistency test relying on a symmetric primitive like one-time signatures or MACs is required. Our scheme can be proved secure under the Bilinear Decisional Diffie-Hellman (BDDH) assumption in pairing groups. Chosen-ciphertext security is obtained at sheer minimal cost. Compared to Waters' IB-KEM our scheme comes with a ciphertext overhead of only one single element whereas computational overhead is one more exponentiation for encryption and one pairing plus two exponentiations for decryption. The security reduction is comparable to the one for Waters' scheme, i.e. it introduces only a small additive component.

Using a chosen-ciphertext secure symmetric encryption scheme (also called a data-encapsulation mechanism DEM) our IB-KEM can be extended to a chosen-ciphertext secure IBE scheme [5]. From a theoretical point of view IB-KEM and IBE are equivalent. However, there are a numerous practical reasons to prefer a IB-KEM over an IBE scheme. The biggest advantage is its flexibility, i.e. an IB-KEM completely decouples the key encapsulation from the asymmetric part. So when performing encryption one is free to pick whatever security parameter necessary without changing the size of the message space. For (standard) public-key encryption the same modular approach is incorporated in many standards due to his simplicity and flexibility (see, e.g., [28, 2]). The same is expected to happen in the new IEEE P1363.3 standard for "Identity-Based Cryptographic Techniques using Pairings" [19].

Our IB-KEM scheme can be extended in a natural way to obtain a chosen-ciphertext secure HIB-KEM with only one additional element in the ciphertext compared to Waters' chosen-plaintext secure HIB-KEM.

A RIGOROUS GAME-BASED PROOF. The proof of Waters' IBE is already quite complex and has many technical parts that we found pretty hard to verify. Additionally, many recent results [11, 13, 23] already use ingredients of Waters' IBE, some more or less in a "black-box" manner which makes verification nearly impossible without having completely understood the original work. This goes along with a general movement in our field to produce proofs that are increasingly hard to verifyour opinion this situation has been getting worse and worse. Our additional components to make Waters IB-KEM chosen-ciphertext secure add even more complexity to the proof.

Motivated by this we give a rigorous, games-based proof of our result that can be easily understood and verified. As an immediate benefit our security reduction achieves some slight improvements over Waters bounds [29]. Unfortunately our proof extends by far the page limit of this extended abstract. The interested reader is referred to [22] for the full details.

1.2 Related Work and Comparison

In this paper we extract the "IB-KEM part" of our pre-print [22] where we furthermore show how our IB-KEM can be extended to the first chosen-ciphertext secure threshold IB-KEM in the standard model. Our technique to obtain the chosen-ciphertext secure IB-KEM is somewhat reminiscent of the method used in [11, 21] to obtain chosen-ciphertext secure *standard encryption*. Interestingly our scheme can be seen as a generalization of the standard public-key encryption scheme from [11], i.e. ignoring all "identity-based components" (and applying some optimizations in the decapsulation algorithm) our scheme can be simplified to exactly their scheme.

In the same work [11] a technique is sketched how to avoid the CHK transformation to get a direct chosen-ciphertext secure IB-KEM construction based on Waters' 2-level HIB-KEM. Compared to our IB-KEM, however, this construction has a weaker (quadratic) security reduction and nearly doubles the public key size. In [22] we carefully review all known chosen-ciphertext secure IB-KEM constructions, including the above proposal, and make an extensive comparison with our scheme.

It turns out that, to the best of our knowledge, our IB-KEM is the most efficient chosen-ciphertext secure IB-KEM scheme in the standard model based on a standard complexity-theoretic assumption.

2 Definitions

2.1 Notation

If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. If S is a set then $s \xleftarrow{\$} S$ denotes the operation of picking an element s of S uniformly at random. We write $\mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \xleftarrow{\$} \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and letting z be the output. We write $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$ and by $z \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$, and letting z be the output.

2.2 Identity Based Key Encapsulation

An *identity-based key-encapsulation mechanism* (IB-KEM) scheme $\mathcal{IBKEM} = (\mathsf{IBKEMkg}, \mathsf{IBKEMkeyder}, \mathsf{IBKEMenc}, \mathsf{IBKEMdec})$ consists of four polynomial-time algorithms. Via $(pk, msk) \xleftarrow{\$} \mathsf{IBKEMkg}(1^k)$ the randomized key-generation algorithm produces master keys for security parameter $k \in \mathbb{N}$; via $sk[id] \xleftarrow{\$} \mathsf{IBKEMkeyder}(msk, id)$ the master computes the secret key for identity id; via $(C, K) \xleftarrow{\$} \mathsf{IBKEMenc}(pk, id)$ a sender creates a random session key K and a corresponding ciphertext C with respect to identity id; via $K \leftarrow \mathsf{IBKEMdec}(sk, C)$

the possessor of secret key sk decapsulates ciphertext C to get back a the session key K. Associated to the scheme is a key space KeySp. For consistency, we require that for all $k \in \mathbb{N}$, all identities id, and all $(C, K) \xleftarrow{\$} \mathsf{IBKEMenc}(pk, id)$, we have $\Pr[\mathsf{IBKEMdec}(\mathsf{IBKEMkeyder}(msk, id), C) = K] = 1$, where the probability is taken over the choice of $(pk, msk) \xleftarrow{\$} \mathsf{IBKEMkg}(1^k)$, and the coins of all the algorithms in the expression above.

Let $\mathcal{IBKEM} = (\mathsf{IBKEMkg}, \mathsf{IBKEMkeyder}, \mathsf{IBKEMenc}, \mathsf{IBKEMdec})$ be an IB-KEM with associated key space KeySp. To an adversary $\mathcal{A}$ we associate the following experiment:

Experiment $\mathbf{Exp}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k)$
$(pk, msk) \xleftarrow{\$} \mathsf{IBKEMkg}(1^k)$
$(id^*, st) \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDer}(\cdot),\textsc{Dec}(\cdot,\cdot)}(\mathtt{find}, pk)$
$K_0^* \xleftarrow{\$} \mathsf{KeySp}\,;\ (C^*, K_1^*) \xleftarrow{\$} \mathsf{IBKEMenc}(pk, id)$
$\gamma \xleftarrow{\$} \{0,1\}\,;\ K^* \leftarrow K_\gamma^*$
$\gamma' \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDer},\textsc{Dec}}(\mathtt{guess}, K^*, C^*, st)$
If $\gamma \neq \gamma'$ then return 0 else return 1

The oracle $\textsc{KeyDer}(id)$ returns $sk[id] \xleftarrow{\$} \textsc{KeyDer}(msk, id)$ with the restriction that $\mathcal{A}$ is not allowed to query oracle $\textsc{KeyDer}(\cdot)$ for the target identity id^*. The oracle $\textsc{Dec}(id, C)$ first computes $sk[id] \xleftarrow{\$} \textsc{KeyDer}(msk, id)$ as above and then returns $K \leftarrow \mathsf{IBKEMdec}(sk[id], id, C)$ with the restriction that in the guess stage $\mathcal{A}$ is not allowed to query oracle $\textsc{Dec}(\cdot,\cdot)$ for the tuple (id^*, C^*). st is some internal state information of adversary $\mathcal{A}$ and can be any (polynomially bounded) string. We define the advantage of $\mathcal{A}$ in the IND-CCA experiment as

$$\mathbf{Adv}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k) \;=\; \left|\Pr\left[\,\mathbf{Exp}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k) = 1\,\right] - \frac{1}{2}\right| .$$

An IB-KEM $\mathcal{IBKEM}$ is said to be *secure against adaptively-chosen ciphertext attacks* if the advantage functions $\mathbf{Adv}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}}(k)$ is a negligible function in k for all polynomial-time adversaries $\mathcal{A}$.

We remark that our security definition is given with respect to "full-identity" attacks, as opposed to the much weaker variant of "selective-identity" attacks where the adversary has to commit to its target identity id^* in advance, even before seeing the public key.

2.3 Target Collision Resistant Hash Functions

Let $\mathcal{F} = (\mathsf{TCR}_s)_{s \in S}$ be a family of hash functions for security parameter k and with seed $s \in S = S(k)$. $\mathcal{F}$ is said to be *collision resistant* if, for a hash function $\mathsf{TCR} = \mathsf{TCR}_s$ (where the seed is chosen at random from S), it is infeasible for an efficient adversary to find two distinct values $x \neq y$ such that $\mathsf{TCR}(x) = \mathsf{TCR}(y)$.

A weaker notion is that of *target collision resistant hash functions*. Here it should be infeasible for an efficient adversary to find, given a randomly chosen

element x and a randomly drawn hash function $\mathsf{TCR} = \mathsf{TCR}_s$, a distinct element $y \neq x$ such that $\mathsf{TCR}(x) = \mathsf{TCR}(y)$. (In collision resistant hash functions the value x may be chosen by the adversary.) Such hash functions are also called *universal one-way hash functions* [24] and can be built from arbitrary one-way functions [24, 25]. We define (slightly informal)

$$\mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) = \Pr[\mathcal{H} \text{ finds a collision in } \mathsf{TCR}].$$

Hash function family TCR is said to be a *target collision resistant* if the advantage function $\mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}$ is a negligible function in k for all polynomial-time adversaries $\mathcal{H}$.

In practice, to build a target collision resistant hash function TCR, one can use a dedicated cryptographic hash function, like SHA-1 [26]. For that reason and to simplify our presentation, in what follows we will consider the hash function TCR to be a fixed function.

3 Assumptions

3.1 Parameter Generation Algorithms for Bilinear Groups

All pairing based schemes will be parameterized by a *pairing parameter generator*. This is a PTA $\mathcal{G}$ that on input 1^k returns the description of an multiplicative cyclic group $\mathbb{G}_1$ of prime order p, where $2^k < p < 2^{k+1}$, the description of a multiplicative cyclic group $\mathbb{G}_T$ of the same order, and a non-degenerate bilinear pairing $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. See [9] for a description of the properties of such pairings. We use $\mathbb{G}_1^*$ to denote $\mathbb{G}_1 \setminus \{0\}$, i.e. the set of all group elements except the neutral element. Throughout the paper we use $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_T, p, \hat{e})$ as shorthand for the description of bilinear groups.

3.2 The BDDH Assumption

Let $\mathcal{PG}$ be the description of pairing groups. Consider the following problem first considered by Joux [20] and later formalized by Boneh and Franklin [9]: Given $(g, g^a, g^b, g^c, W) \in \mathbb{G}_1^4 \times \mathbb{G}_T$ as input, output yes if $W = \hat{e}(g,g)^{abc}$ and no otherwise. More formally, to a parameter generation algorithm for pairing-groups $\mathcal{G}$ and an adversary $\mathcal{B}$ we assotiate the following experiment.

Experiment $\mathbf{Exp}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k)$
$\mathcal{PG} \xleftarrow{\$} \mathcal{G}(1^k)$
$a, b, c, w \xleftarrow{\$} \mathbb{Z}_p^*$
$\beta \xleftarrow{\$} \{0,1\}$
If $\beta = 1$ then $W \leftarrow \hat{e}(g,g)^{abc}$ else $W \leftarrow \hat{e}(g,g)^{w}$
$\beta' \xleftarrow{\$} \mathcal{B}(1^k, \mathcal{PG}, g, g^a, g^b, g^c, W)$
If $\beta \neq \beta'$ then return 0 else return 1

We define the advantage of $\mathcal{B}$ in the above experiment as

$$\mathbf{Adv}^{\mathrm{bddh}}_{\mathcal{G},\mathcal{B}}(k) \;=\; \left|\Pr\left[\,\mathbf{Exp}^{\mathrm{bddh}}_{\mathcal{G},\mathcal{B}}(k)=1\,\right]-\frac{1}{2}\right| .$$

We say that the *Bilinear Decision Diffie-Hellman (BDDH) assumption relative to generator* $\mathcal{G}$ holds if $\mathbf{Adv}^{\mathrm{bddh}}_{\mathcal{G},\mathcal{B}}$ is a negligible function in k for all PTAs $\mathcal{B}$. The BDDH assumption was shown to hold in the generic group model in [7].

4 A Chosen-Ciphertext Secure IB-KEM Based on BDDH

In this section we present our new chosen-ciphertext secure IB-KEM. From now on let $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_T, p, \hat{e}, g)$ be public system parameters obtained by running the group parameter algorithm $\mathcal{G}(1^k)$.

4.1 Waters' Hash Function

We review the hash function $\mathsf{H} : \{0,1\}^n \to \mathbb{G}_1$ used in Waters' identity based encryption schemes [29]. On input of an integer n, the randomized hash key generator $\mathsf{HGen}(\mathbb{G}_1)$ chooses $n+1$ random groups elements $h_0, \ldots, h_n \in \mathbb{G}_1$ and returns $h = (h_0, h_1, \ldots, h_n)$ as the public description of the hash function. The hash function $\mathsf{H} : \{0,1\}^n \to \mathbb{G}_1^*$ is evaluated on a string $id = (id_1, \ldots, id_n) \in \{0,1\}^n$ as the product

$$\mathsf{H}(id) = h_0 \prod_{i=1}^{n} h_i^{id_i}.$$

4.2 The IB-KEM Construction

Let $\mathsf{TCR} : \mathbb{G}_1 \to \mathbb{Z}_p$ be a target collision-resistant hash function (which we assume to be included in the system parameters). Our IB-KEM with identity space $\mathsf{IDSp} = \{0,1\}^n$ ($n = n(k)$) and key space $\mathsf{KeySp} = \mathbb{G}_T$ is depicted in Fig. 1.

A tuple $(g, c_1, u_1^t u_2, c_3)$ is a Diffie-Hellman tuple[1] if $\hat{e}(g, c_3) = \hat{e}(u_1^t u_2, c_1)$. Analogously, $(g, c_1, \mathsf{H}(id), c_2)$ is a Diffie-Hellman tuple if $\hat{e}(g, c_2) = \hat{e}(\mathsf{H}(id), c_1)$. Therefore the check in the decapsulation algorithm $\mathsf{IBKEMdec}$ can be implemented by evaluating the bilinear map four times.

We now show correctness of the scheme, i.e. that the session key K computed in the encapsulation algorithm matches the K computed in the decapsulation algorithm. A correctly generated ciphertext for identity id has the form $C = (c_1, c_2, c_3) = (g^r, \mathsf{H}(id)^r, (u_1^t u_2)^r)$ and therefore $(g, c_1, u_1^t u_2, c_3) = (g, g^r, u_1^t u_2, (u_1^t u_2)^r)$ is always a DH tuple. A correctly generated secret key for identity id has the form $sk[id] = (d_1, d_2) = (\alpha \cdot \mathsf{H}(id)^s, g^s)$. Therefore the decapsulation algorithm computes the session key K as

[1] A tuple $(h, h^a, h^b, h^c) \in \mathbb{G}_1^4$ is said to be a *Diffie-Hellman tuple* if $ab = c \bmod p$.

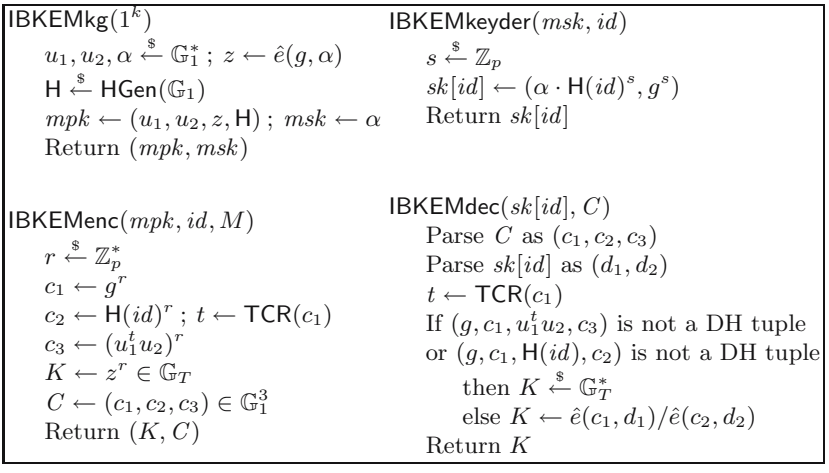


Fig. 1. Our chosen-ciphertext secure IB-KEM

$$\begin{aligned} K &= \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \\ &= \hat{e}(g^r, \alpha \mathsf{H}(id)^s)/\hat{e}(\mathsf{H}(id)^r, g^s) \\ &= \hat{e}(g^r, \alpha) \cdot \hat{e}(g^r, \mathsf{H}(id)^s)/\hat{e}(\mathsf{H}(id)^r, g^s) \\ &= z^r \cdot \hat{e}(g^s, \mathsf{H}(id)^r)/\hat{e}(\mathsf{H}(id)^r, g^s) \\ &= z^r, \end{aligned}$$

as the key computed in the encapsulation algorithm. This shows correctness.

Let $C = (c_1, c_2, c_3) \in \mathbb{G}_1^3$ be a (possibly malformed) ciphertext. Ciphertext C is called *consistent* (w.r.t the public key pk and identity id) if $(g, c_1, u_1^t u_2, c_3)$ and $(g, c_1, \mathsf{H}(id), c_2)$ are Diffie-Hellman tuples, where $t = \mathsf{TCR}(c_1)$. Note that any ciphertext properly generated by the encapsulation algorithm is always consistent. The decapsulation algorithm tests for consistency of the ciphertext. Note that this consistency test can be performed by anybody knowing the public-key. We call this property "public verification" of the ciphertext. In the words of [1] the IB-KEM ciphertext is not *anonymous*.

4.3 More Efficient Decapsulation

We now describe an alternative decapsulation algorithm which is more efficient (but less intuitive). The idea is to make the Diffie-Hellman consistency check implicit in the computation of the key K. This is done by choosing a random values $r_1, r_2 \in \mathbb{Z}_p^*$ and computing the session key as

$$K \leftarrow \frac{\hat{e}(c_1, d_1 \cdot (u_1^t u_2)^{r_1} \cdot \mathsf{H}(id)^{r_2})}{\hat{e}(c_2, d_2 \cdot g^{r_2}) \cdot \hat{e}(g^{r_1}, c_3)} .$$

We claim that this is equivalent to first checking for consistency and then computing the key as $K \leftarrow \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2)$ as in the original decapsulation algorithm.

To prove this claim we define the functions $\Delta_1(C) = \hat{e}(c_1, u_1^t u_2)/\hat{e}(g, c_3)$ and $\Delta_2(C) = \hat{e}(\mathsf{H}(id), c_1)/\hat{e}(g, c_2)$. Then $\Delta_1(C) = \Delta_2(C) = 1$ if and only if C is consistent. Consequently, for random $r_1, r_2 \in \mathbb{Z}_p^*$, $K = \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \cdot (\Delta_1(C))^{r_1} \cdot (\Delta_2(C))^{r_2} \in \mathbb{G}_T^*$ evaluates to $\hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \in \mathbb{G}_T$ if C is consistent and to a random group element otherwise. As in the original decapsulation algorithm. The claim then follows by

$$\begin{aligned} K &= \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \cdot \Delta_1(C)^{r_1} \cdot (\Delta_2(C))^{r_2} \\ &= \hat{e}(c_1, d_1)/\hat{e}(c_2, d_2) \cdot (\hat{e}(c_1, u_1^t u_2)/\hat{e}(g, c_3))^{r_1} \cdot (\hat{e}(\mathsf{H}(id), c_1)/\hat{e}(g, c_2))^{r_2} \\ &= \frac{\hat{e}(c_1, d_1 (u_1^t u_2)^{r_1} \mathsf{H}(id)^{r_2})}{\hat{e}(c_2, d_2 \cdot g^{r_2}) \cdot \hat{e}(g^{r_1}, c_3)} \, . \end{aligned}$$

We remark that the alternative decapsulation algorithm roughly saves two pairing operation (for the cost of a couple of exponentiations).

4.4 Security

Theorem 1. *Assume* TCR *is a target collision resistant hash function. Under the Bilinear Decisional Diffie-Hellman (BDDH) assumption relative to generator* $\mathcal{G}$*, the IB-KEM from Section 4.2 is secure against chosen-ciphertext attacks. In particular, we have*

$$\mathbf{Adv}^{\text{ib-kem-cca}}_{\mathcal{IBKEM},\mathcal{A}} = \mathcal{O}(nq \cdot (\epsilon + q/p) + \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)) \, ,$$

for any IBE adversary $\mathcal{A}$ *running for time* $\mathbf{Time}_{\mathcal{A}}(k) = \mathbf{Time}_{\mathcal{B}} - \Omega(\epsilon^{-2} \cdot \ln(\epsilon^{-1}) + q)$*, where* $\epsilon = \mathbf{Adv}^{\text{bddh}}_{\mathcal{G},\mathcal{B}}(k)$ *and* q *is an upper bound on the number of key derivation/decapsulation queries made by adversary* $\mathcal{A}$*.*

A game-based proof of Theorem 1 can be found in [22]. The proof is mainly based on the one given by Waters [29]. However, we have to do some important modifications to be able to deal with chosen-ciphertext attacks. Furthermore, compared to Waters proof we can achieve a slightly improved security reduction.

Intuitively, security can be best understood by observing that our scheme is a generalization of Waters' (chosen-plaintext secure) IBE scheme, as well as of the chosen-ciphertext secure *public-key* encapsulation scheme from [11]. We remark that unfortunately, there does not seem to be a way to derive security of our IBE scheme directly from security of either of the two schemes and hence details of the whole proof have to be worked out from scratch.

Relation to Waters' IBE scheme. The ciphertext in our scheme is basically identical to the ciphertext from Waters' IBE scheme [29] plus one redundant element (the element c_3) used to check for consistency of the ciphertext. Hence Waters' IBE scheme is obtained by ignoring the computation of c_3 in encapsulation as well as the consistency check in decapsulation.

Relation to the Encryption Scheme from BMW. Clearly, IB-KEM implies (standard) public-key encapsulation by simply ignoring all operations related to the identity. We remark that viewed in this light (i.e. ignoring the

element c_2 in encapsulation/decapsulation and ignoring the key derivation algorithm) our IB-KEM can be simplified to the chosen-ciphertext secure encryption scheme recently proposed by Boyen, Mei, and Waters [11].

5 Extensions

5.1 Chosen-Ciphertext Secure Hierarchical Identity-Based Key Encapsulation

Hierarchical identity-based key encapsulation (HIB-KEM) is a generalization of IB-KEM to identities supporting hierarchical structures [18, 17]. By the relation to Waters IBE scheme it is easy to see that our technique can also be used to make (the KEM variant of) Waters' HIBE chosen-ciphertext secure. To be more precise, we modify Waters' HIB-KEM and add one more element $h_1^{rt}h_2^r$ to the the ciphertext, where t was computed by applying a target-collision hash function to g^r (here r is the randomness used to create the ciphertext). The additional element is used for a consistency check at decryption. The security reduction is exponential in the depth d of the hierarchy, i.e. it introduces, roughly, a multiplicative factor of $(nq)^d$.

5.2 Identity-Based Encryption

Given a IB-KEM and a symmetric encryption scheme, a hybrid identity-based encryption scheme can be obtained by using the IB-KEM to securely transport a random session key that is fed into the symmetric encryption scheme to encrypt the plaintext message. It was recently shown in [5] that if both the IB-KEM and the symmetric encryption scheme are chosen-ciphertext secure, then the resulting hybrid encryption is also chosen-ciphertext secure. The security reduction is tight.

5.3 A Tradeoff Between Public Key Size and Security Reduction

As independently discovered in [13, 23], there exists an interesting trade-off between key-size of Waters' hash H and the security reduction of the IBE scheme.

The construction modifies Waters hash H as follows: Let the integer $l = l(k)$ be a new parameter of the scheme. In particular, we represent an identity $id \in \{0,1\}^n$ as an n/l-dimensional vector $id = (id_1, \ldots, id_{n/l})$, where each id_i is an l bit string. Waters hash is then redefined to $\mathsf{H} : \{0,1\}^n \to \mathbb{G}_1$, with $\mathsf{H}(id) = h_0 \prod_{i=1}^{n/l} h_i^{id_i}$ for random public elements $h_0, h_1, \ldots, h_{n/l} \in \mathbb{G}_1$. Waters' original hash function is obtained as the special case $l = 1$. It is easy to see that using this modification in our IBE scheme (i) reduces the size of the public key from $n+4$ to $n/l+4$ group elements, whereas (ii) it adds another multiplicative factor of 2^l to the security reduction of the IBE scheme (Theorem 1).

5.4 Selective-Identity Chosen-Ciphertext Secure IB-KEM

For the definition of a selective-identity chosen-ciphertext secure IB-KEM we change the security experiment such that the adversary has to commit to the

target idenity id^* before seeing the public key. Clearly, this is a weaker security requirement. We quickly note that (using an algebraic technique from [6]) by replacing Waters' hash H with $\mathsf{H}(id) = h_0 \cdot h_1^{id}$ (for $id \in \mathbb{Z}_p$) we get a selective-id chosen-ciphertext secure IB-KEM. Note that the size of the public-key of this scheme drops to 3 elements.

5.5 Implementing the Collision Resistant Hash Function TCR

In practice, to build a target collision resistant hash function, one can use a dedicated cryptographic hash function, like SHA-1 [26].

Every injective function $\mathsf{TCR} : \mathbb{G}_1 \to \mathbb{Z}_p$ trivially also is (target) collision resistant (with zero advantage). Boyen, Mei and Waters [11] note that for bilinear maps defined on elliptic curves there exists a very efficient way to implement such injective mappings. We refer to [11] for more details.

References

1. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In V. Shoup, editor, *CRYPTO 2005*, LNCS. Springer-Verlag, Aug. 2005.
2. American National Standards Institute (ANSI) X9.F1 subcommittee. ANSI X9.63 Public key cryptography for the Financial Services Industry: Elliptic curve key agreement and key transport schemes, July 5, 1998. Working draft version 2.0.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer-Verlag, Aug. 1998.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
5. K. Bentahar, P. Farshim, J. Malone-Lee, and N. Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058, 2005. `http://eprint.iacr.org/`.
6. D. Boneh and X. Boyen. Efficient selective-id secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, May 2004.
7. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer-Verlag, May 2005.
8. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, Aug. 2001.
9. D. Boneh and M. K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
10. D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In A. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 87–103. Springer-Verlag, Feb. 2005.

11. X. Boyen, Q. Mei, and B. Waters. Simple and efficient CCA2 security from IBE techniques. In *ACM Conference on Computer and Communications Security—CCS 2005*, pages 320–329. New-York: ACM Press, 2005. Available at `http://eprint.iacr.org/2005/288/`, August 2005.
12. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.
13. S. Chatterjee and P. Sarkar. Trading time for space: Towards an efficient ibe scheme with short(er) public parameters in the standard model. Proceedings of ICISC, to appear, 2005.
14. C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363, Cirencester, UK, Dec. 17–19, 2001. Springer-Verlag.
15. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer-Verlag, Aug. 1999.
16. D. Galindo and I. Hasuo. Security notions for identity based encryption. Cryptology ePrint Archive, Report 2005/253, 2005. `http://eprint.iacr.org/`.
17. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer-Verlag, Dec. 2002.
18. J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466–481. Springer-Verlag, Apr. 2002.
19. IEEE P1363.3 Committee. IEEE 1363.3 / CfS — standard for identity-based cryptographic techniques using pairings. `http://grouper.ieee.org/groups/1363/index.html/`, Feb. 2006. Call for submissions.
20. A. Joux. A one round protocol for tripartite diffie-hellman. In *Algorithmic Number Theory – ANTS IV*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.
21. E. Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *TCC 2006*, LNCS, pages 581–600. Springer-Verlag, Mar. 2006.
22. E. Kiltz and D. Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles, Jan. 2006. Available at `http://eprint.iacr.org/2006/034/`.
23. D. Naccache. Secure and *practical* identity-based encryption. Cryptology ePrint Archive, Report 2005/369, 2005. `http://eprint.iacr.org/`.
24. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989.
25. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.
26. Secure hash standard. National Institute of Standards and Technology, NIST FIPS PUB 180-1, U.S. Department of Commerce, Apr. 1995.
27. A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, Aug. 1985.
28. V. Shoup. A proposal for an ISO standard for public key encryption (version 2.1). manuscript, 2001. Available on `http://shoup.net/papers/`.
29. B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer-Verlag, May 2005.

Generic Transforms to Acquire CCA-Security for Identity Based Encryption: The Cases of FOPKC and REACT

Takashi Kitagawa[1], Peng Yang[2], Goichiro Hanaoka[1], Rui Zhang[1], Hajime Watanabe[1], Kanta Matsuura[2], and Hideki Imai[1]

[1] Research Centre for Information Security (RCIS), National Institute of Advanced Industrial Science and Technology (AIST) Akihabara-Daibiru Room 1102, 1-18-13 Sotokanda, Chiyoda-ku, Tokyo, 101-0021, Japan
{t-kitagawa, hanaoka-goichiro, r-zhang, h-watanabe, h-imai}@aist.go.jp

[2] Institute of Industrial Science, University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, Japan
{pengyang, kanta}@iis.u-tokyo.ac.jp

Abstract. Fujisaki-Okamoto (FOPKC) conversion [13] and REACT[17] are widely known to be able to generically convert a weak public key encryption scheme to a strong encryption scheme. In this paper, we discuss applications of FOPKC conversion and REACT to Identity Based Encryptions (IBE). It has not been formally verified yet that whether these conversions are generic in the IBE setting.

Our results show that both conversions are effective in the IBE case: plain REACT already achieves a good security reduction while the plain FOPKC conversion results in bad running time of the simulator. We further propose a simple modification to the plain FOPKC that solves this problem. Finally, we choose some concrete parameters to explain (visually) the effect of how the modified FOPKC substantially improves reduction cost regarding the plain conversion.

1 Introduction

Identity based encryption (IBE) [18] is a public key encryption scheme where the encryption key can be an arbitrary string, such as the recipient's identity, thus the distribution of public key certificates can be avoided. However, only recently did the first practical full-functional IBE schemes [7, 9] came into birth, which are proven secure in the random oracle model [10, 3]. Subsequent research further extended to the standard model [8, 5, 4, 19].

It has been shown [1] that the strongest security notion for IBE is *indistinguishability against adaptive chosen* ID *and adaptive chosen ciphertext attacks* (IND-ID-CCA). Nevertheless, in the random oracle model, many IND-ID-CCA IBE schemes (like [7, 14]) are often built with the following strategy:Full Scheme = Basic Scheme + Transform where the basic scheme is the one appeared in [6, 7] and the transform may be those proposed previously for public key encryptions,

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 348–359, 2006.
© Springer-Verlag Berlin Heidelberg 2006

like FOCRYPTO [12], FOPKC [13] and REACT [17]. We have confirmed that the FOCRYPTO conversion is generic for any CPA-secure IBE [20]. Here, we would like to consider the FOPKC and REACT conversions.

Related Work. As mentioned already, Boneh and Franklin proposed the first practical IBE scheme which became the base for many related researches. Their proof was quite loose, however, nobody had considered how to improve the security reduction cost of their proof, even nobody had challenged the correctness of their proof until recently, Galindo [14] noticed a small flawed step in the proof of Boneh-Franklin's paper, however, the reduction in Galindo's corrected proof was even looser. We also note that in fact, the proof given in [7, 14] did not take account of applying generic FO transforms but has mainly considered how to reduce the security of the "full" scheme to that of an IND-CCA public key encryption scheme.

Another variant of Boneh-Franklin IBE scheme with tighter security reduction was given by Libert and Quisquater [16], with a REACT-like appearance.

Our Results. We prove that these conversions (FOPKC and REACT) can be applied to IBE generically with good reduction costs. More precisely, applying the plain FOPKC to Boneh-Franklin's basic scheme, one immediately acquires the scheme proposed recently by Galindo [14], yet with a very loose security reduction. Recall that in the public key setting, the FOPKC conversion can be proven with a "tight" security reduction to its underlying primitives.

However, we can partially overcome this problem with a tiny modification to the plain FOPKC. The modification is itself very simple and computationally efficient: just hash the ID with other inputs to the random oracle. However, this simple idea actually works! The modified FOPKC conversion admits exactly tight reduction reduction as its public key counterpart. On the other hand, the plain REACT already gives a good reduction cost, without any modification. Interestingly, these results may indicate a separation between the chosen plaintext attack (CPA) and plaintext checking attack (PCA) in the IBE setting.

We further choose some concrete parameters to explain how the modified FOPKC improves the reduction cost, by estimating the average running time of the simulator. For chosen parameters, using a single PC (or a single dedicated hardware), an IND-ID-CCA adversary breaks the IND-ID-CCA security of "the basic Boneh-Franklin scheme + the plain FOPKC conversion" with about 10^{24} years in addition to break the IND-ID-CPA security of the basic Boneh-Franklin scheme. However, it needs only additional 10^8 or 10^9 years in the case of the modified FOPKC conversion. Consider possible paralleled computing, say 1 million PCs, this becomes $10^2 \sim 10^3$ years.

Remark 1. *Besides the* IND-ID-CCA *security, there is an another weaker security notion for IBE, called security against* selective ID *(*sID*) attack [8]. This security notion is also very useful and has applications in constructing CCA-secure public key encryptions. However, with similar discussions of this paper, one can reach similar results for* sID *secure IBE: the FO*PKC *(REACT) are also*

feasible to upgrade weak sID *secure IBE to acquire* CCA *security, and the reduction costs follow naturally the discussions above.*

2 Preliminary

In this section we review the definition of IBE, several conventions and security notions.

Identity Based Encryption. Formally, an IBE scheme $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ consists of the four algorithms.

- $\mathcal{S}$, the setup algorithm, takes as input security parameter $k \in \mathbb{Z}$, and outputs system parameters params and the master-key master-key. params has a description of a message space MSPC and a ciphertext space CSPC.
- $\mathcal{X}$, the extraction algorithm, takes as input params, master-key and an arbitrary $\mathsf{ID} \in \{0,1\}^*$, and outputs a private key d. ID is an arbitrary string and it is used as a public key. d is the corresponding private key.
- $\mathcal{E}$, the encryption algorithm, takes as input params, ID and $M \in$ MSPC, and outputs a ciphertext $C \in$ CSPC.
- $\mathcal{D}$, the decryption algorithm takes as input params, a ciphertext $C \in$ CSPC and a private key d, and outputs the corresponding plaintext $M \in$ MSPC.

IND-ID-CCA **Security and** IND-ID-CPA **Security.** Boneh and Franklin [7] defined the indistinguishability (IND) for IBE schemes. An IBE which is indistinguishable against adaptive chosen identity and adaptive chosen ciphertext attack (IND-ID-CCA) has the strongest security and one which is indistinguishable against adaptive chosen identity and chosen plaintext attack (IND-ID-CPA) has even weaker security.

In their model, the two security notions were defined in the following games.

Setup: The challenger takes a security parameter k and runs $\mathcal{S}$. It gives the adversary params and keeps the master-key to itself.

Phase 1: The adversary issues queries $q_1, \cdots, q_m$ adaptively, where query q_i is one of: Extraction query or Decryption query. For the detail of these queries, please see [6].

Challenge: Once the adversary decides that Phase 1 is over it outputs a pair of messages $M_0, M_1 \in$ MSPC of equal length and an ID^* on which it wishes to be challenged. ID^* must not have appeared in any Extraction query in Phase 1. The challenger picks a random bit $b \in \{0,1\}$ and sets $C^* = \mathcal{E}(\mathsf{params}, \mathsf{ID}^*, M_b)$. It sends C^* to the adversary.

Phase 2: The adversary issues more queries $q_{m+1}, \cdots, q_n$, where query q_i is as in Phase 1. In Phase 2, the adversary cannot ask about neither ID^* nor C^*.

Guess: Finally, the adversary outputs a result $b' \in \{0,1\}$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. The advantage of IND-ID-CCA adversary $\mathcal{A}$ is defined as follows: $Adv_{\mathcal{A}}(k) = |\Pr[b = b'] - 1/2|$. The probability is over the random bits used by the challenger and the adversary.

Definition 1 (IND-ID-CCA). *We say that an IBE scheme is secure in the sense of* IND-ID-CCA *if $Adv_{\mathcal{A}}$ is negligible for any polynomial time algorithm $\mathcal{A}$.*

IND-ID-CPA **Game.** This game is similar to but easier than IND-ID-CCA game except that the adversary $\mathcal{A}$ is more limited. $\mathcal{A}$ cannot make decryption queries to the challenger. The advantage of IND-ID-CPA adversary $\mathcal{A}$ is also defined as follows: $Adv_{\mathcal{A}}(k) = |\Pr[b = b'] - 1/2|$. The probability is over the random bits used by the challenger and the adversary.

Definition 2 (IND-ID-CPA). *We say that an IBE scheme is secure in the sense of* IND-ID-CPA *if $Adv_{\mathcal{A}}$ is negligible for any polynomial time algorithm $\mathcal{A}$.*

OW-ID-PCA **Security.** Roughly speaking, the security goal, onewayness (OW), means that if given the ciphertext of a random plaintext the adversary cannot produce the plaintext in its entirety. In the plaintext checking attack model, the adversary can access to the PC(plaintext checking) oracle which takes as input a public key ID, a plaintext M and a ciphertext C and outputs "yes" or "no" whether C is the ciphertext of M. Formally, OW-ID-PCA security is defined by OW-ID-PCA Game. Due to the space limitation, we omit the formal definition of OW-ID-PCA game.

Definition 3 (OW-ID-PCA). *We say that an IBE scheme is secure in the sense of* OW-ID-PCA *$Adv_{\mathcal{A}}$ is negligible for any polynomial time algorithm $\mathcal{A}$.*

3 The FO$_{\text{PKC}}$ Conversion for IBE

It is absorbing to find out an authentic way to enhance weak IBE schemes to strongly secure ones. In the conventional public key cryptosystems, there exist some such conversions, and the FO$_{\text{PKC}}$ [11, 13] is an example, besides, it is very efficient and achieves a tight reduction cost. Since an IBE is a different primitive from traditional public key encryption, one may think these conversions are not immediately a solution for IBE.

In this section, we investigate the FO$_{\text{PKC}}$ for IBE. More interesting result is that although a most significant merit of the FO$_{\text{PKC}}$ is its tight reduction cost for conventional public key encryption schemes, however, this merit could not hold if we apply the FO$_{\text{PKC}}$ *straightforwardly* into IBE case. In order to solve the problem, we slightly revise the plain FO$_{\text{PKC}}$. The modification is quite simple in shape, but the right thing that we need.

3.1 The Plain FO$_{\text{PKC}}$

Let $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ be an IND-ID-CPA IBE. Then, we can construct an another IBE $\Pi_1 = \{\mathcal{S}_1, \mathcal{X}_1, \mathcal{E}_1, \mathcal{D}_1\}$ as follows: Let l_1 be a bit length of a plaintext of Π, l_2 be a bit length of a plaintext of Π_1 and COIN(k) be Π's coin-flipping space. The conversion is constructed in Table 1.

Table 1. The Plain FOPKC

The Plain FOPKC
Setup $\mathcal{S}_1$: It is as $\mathcal{S}$. In addition, it picks a hash function H.
Extraction $\mathcal{X}_1$: It is as $\mathcal{X}$.
Encryption $\mathcal{E}_1$: It takes a system parameter params, an encryption key ID and a message M. $\mathcal{E}_1(\mathsf{params}, \mathsf{ID}, M; \sigma) = \mathcal{E}\big(\mathsf{params}, \mathsf{ID}, M\|\sigma; H(M, \sigma)\big)$, where σ is a randomly chosen $l_1 - l_2$ bit string.
Decryption $\mathcal{D}_1$: Let C be a ciphertext to decrypt. 1. Computes $\mathcal{D}(\mathsf{params}, d, C) = M'$ and let $[M']^{l_2} = M$ and $[M']_{l_1-l_2} = \sigma$ where $[a]^b$ and $[a]_b$ denote the first and the last b bits of a string a, respectively. 2. Tests that $\mathcal{E}\big(\mathsf{params}, \mathsf{ID}, M||\sigma; H(M, \sigma)\big) = C$. If not, outputs "reject". 3. Outputs M as the decryption of C.

Theorem 1. *Suppose the hash function H is the random oracle and Π is a γ-uniform IBE encryption scheme. Let $\mathcal{B}$ be an* IND-ID-CCA *adversary which has advantage $\epsilon(k)$ against Π_1 and it runs in time at most $t(k)$. Suppose $\mathcal{B}$ makes at most q_H H queries, q_E Extraction queries and q_D Decryption queries. Suppose that executing $\mathcal{E}$ once needs at most time τ Then there is an* IND-ID-CPA *adversary $\mathcal{A}$ which has advantage at least $(\epsilon(k) + (1/2) - q_H/(2^{l_1-l_2}))(1 - q_D\gamma) - (1/2)$ against Π. Its running time is $t(k) + q_H \cdot q_D \cdot \tau$.*

Proof. We show how to construct adversary $\mathcal{A}$ by using adversary $\mathcal{B}$ as an oracle. The challenger starts an IND-ID-CPA game by executing $\mathcal{S}$ and generates params and $\mathsf{master\text{-}key}$. $\mathcal{A}$ works by interacting with $\mathcal{B}$ in an IND-ID-CCA game as follows:

Setup: $\mathcal{A}$ gives params to $\mathcal{B}$.

Phase 1: Three sorts of queries are answered as follows:

- **H-queries:** $\mathcal{A}$ maintains a list of tuples $\langle M_i, \sigma_i, h_i\rangle$ as explained below. We refer to this list as the H^{list}. The list is initially empty. When $\mathcal{B}$ queries $H(M_i, \sigma_i)$, $\mathcal{A}$ responds as follows:
  1. If the query M_i, σ_i already appears on the H^{list} in a tuple $\langle M, \sigma_i, h_i\rangle$ then $\mathcal{A}$ responds with $H(M_i, \sigma_i) = h_i$.
  2. Otherwise, $\mathcal{A}$ picks a random element h_i from $\mathsf{COIN}(k)$ of Π.
  3. $\mathcal{A}$ adds the tuple $\langle M_i, \sigma_i, h_i\rangle$ to the H^{list} and returns h_i.
- **Extraction queries:** Let $\langle \mathsf{ID}_i\rangle$ be an Extraction query issued by $\mathcal{B}$. $\mathcal{A}$ responds the corresponding decryption key by using his extraction oracle.
- **Decryption queries:** Let $\langle \mathsf{ID}_i, C_i\rangle$ be a Decryption query issued by $\mathcal{B}$. $\mathcal{A}$ responds as follows:
  1. Find a pair of tuples $\langle M, \sigma, h\rangle$ from the H^{list}, such that $\mathcal{E}(\mathsf{params}, \mathsf{ID}_i, M\|\sigma; h) = C_i$.
  2. Outputs M if there exists such a pair of tuples, or outputs "reject" otherwise.

Challenge: Once $\mathcal{B}$ decides that Phase 1 is over it outputs a public key ID^* ($\mathsf{ID}^* \neq \mathsf{ID}_i$) and two messages M_0, M_1 on which it wishes to be challenged.

$\mathcal{A}$ randomly chooses two $l_1 - l_2$ bit strings σ_0 and σ_1 and sends $\langle \mathsf{ID}^*, M_0\|\sigma_0, M_1\|\sigma_1\rangle$ to the challenger.
The challenger picks a random bit b and sets $C = \mathcal{E}(\mathsf{params}, \mathsf{ID}^*, M_b||\sigma_b)$. Then $\mathcal{A}$ gives C as the challenge to $\mathcal{B}$.

Phase 2: Three sorts of queries are answered as the same as in Phase 1.

Guess: Once $\mathcal{B}$ decides that Phase 2 is over it outputs a guess b'.

After $\mathcal{B}$ outputs the guess b', $\mathcal{A}$ outputs this bit b' as his answer.

In order to calculate the reduction cost, we first define the five events as follows:

- F: $\mathcal{A}$ fails to answer a decryption query at some point during the game.
- SA: $\mathcal{A}$ fails to answer a decryption query at some point during the game.
- SB: $\mathcal{B}$ wins the IND-ID-CCA game in the case that event F does not occur.
- $\mathsf{A0}$: $\mathcal{B}$ queries $H(M_b, \sigma_b)$.
- $\mathsf{A1}$: $\mathcal{B}$ queries $H(M_{\bar{b}}, \sigma_{\bar{b}})$.

Then, we have, $\Pr[\mathsf{SB}] = \Pr[\mathsf{SB}|\mathsf{A0}]\Pr[\mathsf{A0}] + \Pr[\mathsf{SB}|\neg\mathsf{A0} \wedge \mathsf{A1}]\Pr[\neg\mathsf{A0} \wedge \mathsf{A1}] + \Pr[\mathsf{SB}|\neg\mathsf{A0} \wedge \neg\mathsf{A1}]\Pr[\neg\mathsf{A0} \wedge \neg\mathsf{A1}]$
and $\Pr[\mathsf{SA}] = \Pr[\mathsf{SA}|\mathsf{A0}]\Pr[\mathsf{A0}] + \Pr[\mathsf{SA}|\neg\mathsf{A0} \wedge \mathsf{A1}]\Pr[\neg\mathsf{A0} \wedge \mathsf{A1}] + \Pr[\mathsf{SA}|\neg\mathsf{A0} \wedge \neg\mathsf{A1}]\Pr[\neg\mathsf{A0} \wedge \neg\mathsf{A1}]$.

From the specification of $\mathcal{A}$, the following equations holds: $\Pr[\mathsf{SA}|\mathsf{A0}] = 1$, $\Pr[\mathsf{SA}|\neg\mathsf{A0} \wedge \mathsf{A1}] = 0$ and $\Pr[\mathsf{SB}|\neg\mathsf{A0} \wedge \neg\mathsf{A1}] = \Pr[\mathsf{SA}|\neg\mathsf{A0} \wedge \neg\mathsf{A1}]$.

Thus, we have, $\Pr[\mathsf{SA}] - \Pr[\mathsf{SB}] = (1 - \Pr[\mathsf{SB}|\mathsf{A0}])\Pr[\mathsf{A0}] - \Pr[\mathsf{SB}|\neg\mathsf{A0} \wedge \neg\mathsf{A1}]\Pr[\neg\mathsf{A0} \wedge \neg\mathsf{A1}] \geq -\Pr[\neg\mathsf{A0} \wedge \mathsf{A1}]$.

Since $\Pr[\neg\mathsf{A0} \wedge \mathsf{A1}] \leq q_H/2^{l_1-l_2}$, we have $\Pr[\mathsf{SA}] \geq \epsilon + (1/2) - q_H/2^{l_1-l_2}$.

Next, we estimate $\Pr[\neg\mathsf{F}]$. The event F occurs only when $\mathcal{B}$ submits a Decryption query $\langle \mathsf{ID}, C\rangle$ such that $C = \mathcal{E}\big(\mathsf{params}, \mathsf{ID}, M\|\sigma; H(M, \sigma)\big)$ without asking $H(M, \sigma)$. This case happens with probability at most γ , and therefore, we have that $\Pr[\neg\mathsf{F}] \leq (1-\gamma)^{q_D} \simeq 1 - q_D\gamma$.

Hence, we have that $Adv_{\mathcal{A}}(k) \geq (\epsilon + \frac{1}{2} - \frac{q_H}{2^{l_1-l_2}})(1 - q_D\gamma) - 1/2$.

Finally, we estimate $\mathcal{A}$'s running time. Since in addition to $\mathcal{B}$'s running time, $\mathcal{A}$ has to run $\mathcal{E}$ for q_H times for responding to each Decryption query, $\mathcal{A}$'s running time is estimated as $t(k) + q_H \cdot q_D \cdot \tau$. □

Discussion: Running Time of $\mathcal{A}$. Bellare and Rogaway [3] proposed the notion of *exact security*, which says, a reduction is meaningful, if given an adversary $\mathcal{B}$ against Π, an adversary $\mathcal{A}$ can be constructed with essentially the same amount of time and success probability against Π_1. Now we focus on the running times of $\mathcal{A}$ and $\mathcal{B}$. As shown in Theorem 1, there exists a polynomial time reduction $\mathcal{B}$ to $\mathcal{A}$: in the reduction given above $\mathcal{A}$'s running time is estimated as $t(k) + q_H \cdot q_D \cdot \tau$ where $t(k)$ is $\mathcal{B}$'s running time. Assuming that q_H and q_D are estimated as 2^{60} and 2^{40} respectively, $\mathcal{A}$ has to run $\mathcal{E}$ for 2^{100} times, which are computationally *infeasible* in practice. (Notice that a Decryption query requires on-line computation, while a G-query only requires off-line hash computation.)

3.2 The Modified FOPKC

Since for the plain FOPKC, the adversary $\mathcal{A}$ cannot determine the ciphertext when it is asked a hash query, it has to do re-encryption for all the hash queries in the list to extract the plaintext. This is essentially why the reduction cost was bad. Here we shall slightly modify the plain FOPKC: The idea is very simple: just include the ID as part of a hash query. Though it is very simple, we shall go on to show it actually solve the above problem.

Let $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ be an IBE scheme which is secure in the sense of IND-ID-CPA. We denote the new encryption scheme as $\Pi_2 = \{\mathcal{S}_2, \mathcal{X}_2, \mathcal{E}_2, \mathcal{D}_2\}$. The modified conversion is almost same as plain conversion. Thus we omit to describe the formal definition of the modified conversion. We will describe it in the full paper version.

Theorem 2. *Suppose that the hash function H is the random oracle and Π is γ-uniform IBE encryption scheme. Let $\mathcal{B}$ be an* IND-ID-CCA *adversary which has advantage $\epsilon(k)$ against Π_2 and it runs in time at most $t(k)$. Suppose $\mathcal{B}$ makes at most q_H H-queries, q_E Extraction queries and q_D Decryption queries. Suppose that encrypting one message needs time τ. Then there is an* IND-ID-CPA *adversary $\mathcal{A}$ which has advantage at least $(\epsilon(k) + (1/2) - q_H/(2^{l_1 - l_2}))(1 - q_D\gamma) - 1/2$ against Π.Its running time is $t(k) + q_H \cdot \tau$*

Proof. Similar strategy of the proof of Theorem 1 applies here, i.e., construct IND-ID-CPA adversary $\mathcal{A}$ for Π by using IND-ID-CCA adversary $\mathcal{B}$ for Π_2 as an oracle. The construction of $\mathcal{A}$ is almost same in Theorem 1. The main difference is how to answer Decryption queries. Due to space limitation, we denote only how to answer Decryption queries.

$\mathcal{A}$ answers Decryption queries as follows: Let $\langle \mathsf{ID}_i, C_i \rangle$ be a decryption query issued by $\mathcal{B}$.

1. Finds a tuple $\langle \sigma_j, M_j, \mathsf{ID}_j, g_j, C_j \rangle$ from the H^{list} such that $\mathsf{ID}_i = \mathsf{ID}_j$ and $C_i = C_j$.
2. Outputs M_j if there exists such a tuple, or outputs "reject" otherwise.

The advantage of $\mathcal{A}$ can be evaluate in the same way as in Theorem 1, so we omit to describe the detail of the evaluation here.

Finally, we estimate the running time of $\mathcal{A}$. Since adding the running time of $\mathcal{B}$, $\mathcal{A}$ has to run $\mathcal{E}$ once when a new H-query is asked. Therefore, $\mathcal{A}$'s running time is estimated as $t(k) + q_H \cdot \tau$.

3.3 Comparison: Running Time of $\mathcal{A}$ in the Plain FOPKC and the Modified FOPKC

We compare the running times of simulators for Π_1 and Π_2. In this comparison, we especially focus on times to run the encryption algorithm $\mathcal{E}$ which is required for each simulation. It is believed that if a simulator has to run $\mathcal{E}$ for more than 2^{80} times, then it does not properly work in a realistic time. Now, we have that

$$\#\mathcal{E}(\Pi_1)(\sim 2^{100}) \gg 2^{80} \gg \#\mathcal{E}(\Pi_2)(\sim 2^{60})$$

where $\#\mathcal{E}(\cdot)$ denotes the times to run $\mathcal{E}$ in the simulation. This implies that the running time of the simulator for Π_2 is considered realistic.

4 REACT for IBE

REACT[17] was originally designed for (OW-PCA secure) public key cryptosystems, to have CCA security. Again, it was not known if REACT can be applied to IBE generically. We investigate the fact in this section. Interestingly plain REACT is not only effective for IBE, but also gives a tight reduction cost, as it does for traditional public key cryptosystems.

4.1 The Plain REACT for IBE

Let $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ be an OW-ID-PCA IBE. Let MSPC be a message space of Π and CSPC be a ciphertext space of Π. Then we can construct another IBE $\Pi_3 = \{\mathcal{S}_3, \mathcal{X}_3, \mathcal{E}_3, \mathcal{D}_3\}$ which is secure against IND-ID-CCA. Let MSPC′ and CSPC′ be a message space and a ciphertext space of Π_3. A ciphertext of Π_3 consists of three components c_1, c_2 and c_3. We denote the bit length of these components l_1, l_2 and l_3 respectively. The definition of Π_3 is as follows in Table 2.

Table 2. The Plain REACT for IBE

The Plain REACT for IBE
Setup $\mathcal{S}_3$: It is as $\mathcal{S}$. In addition it picks two hash functions: $G : \mathsf{MSPC} \to \{0,1\}^{l_2}, H : \mathsf{MSPC} \times \mathsf{MSPC}' \times \{0,1\}^{l_1} \times \{0,1\}^{l_2} \to \{0,1\}^{l_3}$.
Extraction $\mathcal{X}_3$: It is as $\mathcal{X}$.
Encryption $\mathcal{E}_3$: For a massage M and random values R, it gets $c_1 = \mathcal{E}(\mathsf{params}, \mathsf{ID}, R; r)$, $c_2 = G(R) \oplus M, c_3 = H(R, M, c_1, c_2)$. The ciphertext consists of the triple $C = (c_1, c_2, c_3)$.
Decryption $\mathcal{D}_3$: We assume that the ciphertext to decrypt is $C = (c_1, c_2, c_3)$. First it decrypts c_1 and gets R. Then it computes $K = G(R)$ and $M = c_2 \oplus K$. It returns M if $c_3 = H(R, M, c_1, c_2)$. Otherwise , it outputs "Reject".

Theorem 3. *Suppose the hash functions are random oracles. Let $\mathcal{B}$ be an* IND-ID-CCA *adversary with advantage $\epsilon(k)$ against Π_3 and its running time is $t(k)$. Suppose $\mathcal{B}$ makes at most q_G G-queries, q_H H-queries, q_E extraction queries and q_D decryption queries. Then there is an* OW-ID-PCA *adversary $\mathcal{A}$ which has advantage $2\epsilon(k) - q_D(\frac{1}{2^{l_2}} + \frac{1}{2^{l_3}})$. Its running time is $t(k) + (q_G + q_H) \cdot O(1)$.*

Proof. We show how to construct adversary $\mathcal{A}$ by using adversary $\mathcal{B}$ as an oracle. The challenger starts an OW-ID-PCA game by executing $\mathcal{S}$. Then $\mathcal{A}$ works by interacting with $\mathcal{B}$ in an IND-ID-CCA game as follows:

Setup: $\mathcal{A}$ gives params to $\mathcal{B}$.

Phase 1: Four sorts of queries are answered as follows:

G-queries and H-queries: For each queries, $\mathcal{A}$ maintains G^{list} and H^{list} and responds queries.

Extraction queries: A responds by using his own extraction oracle.

Decryption queries: Let $\langle \mathsf{ID}_i, c_1, c_2, c_3 \rangle$ be a Decryption query issued by $\mathcal{B}$. $\mathcal{A}$ responds as follows:

1. $\mathcal{A}$ picks up a tuple $\langle R', M', c_1', c_2', c_3' \rangle$ from H^{list} such that $c_3 = c_3'$.
2. $\mathcal{A}$ computes $K' = G(R')$.
3. Checks if $c_2 = M' \oplus K'$. If this holds, $\mathcal{A}$ queries $\langle \mathsf{ID}_i, R', c_1 \rangle$ to the PC oracle.
4. If the PC oracle answers "yes", $\mathcal{A}$ returns M' to $\mathcal{B}$. Otherwise, $\mathcal{A}$ outputs "reject".

Challenge: Once $\mathcal{B}$ decides that Phase 1 is over it outputs a public key ID^* ($\mathsf{ID}^* \neq \mathsf{ID}_i$) and two message M_0, M_1 on which it wishes to be challenged. $\mathcal{A}$ sends ID^* to the challenger and receives a ciphertext C^*. $\mathcal{A}$ generates a l_2 bit random string c_2 and a l_3 bit random string c_3. $\mathcal{A}$ gives $\langle C^*, c_2, c_3 \rangle$ to $\mathcal{B}$ as a challenge ciphertext.

Phase 2: Four sorts of queries are answered as the same as in Phase 1.

Guess: Once $\mathcal{B}$ decides that Phase 2 is over it outputs a guess b'.

After $\mathcal{B}$ outputs a guess b', $\mathcal{A}$ picks all Rs which appear in tuples on the G^{list} and the H^{list}. For each R, $\mathcal{A}$ queries $\langle \mathsf{ID}^*, R, C^* \rangle$ to PC oracle. If PC oracle returns "yes", $\mathcal{A}$ outputs the R as the answer of $\mathsf{OW\text{-}ID\text{-}PCA}$ game.

To estimate the advantage of $\mathcal{A}$, we define the following four events:

- SA: $\mathcal{A}$ wins the $\mathsf{OW\text{-}ID\text{-}PCA}$ game
- SB: $\mathcal{B}$ wins the $\mathsf{IND\text{-}ID\text{-}CCA}$ game
- AB: $\mathcal{B}$ asks a query for $G(R^*)$ or $H(R^*, M_b, c_1, c_2)$ at some point during the game
- F: the simulation fails before the event AB occurs

Then we take the same discussion as in the proof of Theorem 1 and we have that $\Pr[\mathsf{SB}|\neg\mathsf{F}]\Pr[\neg\mathsf{F}] \geq \epsilon(k) + (1/2) - \Pr[\mathsf{F}]$.

Since $\Pr[\mathsf{SB}|\neg\mathsf{F} \wedge \neg\mathsf{AB}] = 1/2$, we also have $\Pr[\mathsf{SB}|\neg\mathsf{F}] = \Pr[\mathsf{SB}|\neg\mathsf{F} \wedge \mathsf{AB}]\Pr[\mathsf{AB}] + (1/2)(1 - \Pr[\mathsf{AB}]) \leq (1/2)\Pr[\mathsf{AB}] + 1/2$. Hence, we have $((1/2)\cdot\Pr[\mathsf{AB}] + 1/2)\Pr[\neg\mathsf{F}] \geq \epsilon(k) + (1/2) - \Pr[\mathsf{F}]$, and therefore, $\Pr[\mathsf{AB}] \geq 2\epsilon(k) - \Pr[\mathsf{F}]$.

Next, we estimate $\Pr[\mathsf{F}]$. The event F occurs only in either (1) $\mathcal{B}$ submits a Decryption query $\langle \mathsf{ID}, c_1, G(R) \oplus M, c_3 \rangle$ such that $c_1 = \mathcal{E}(\mathsf{params}, \mathsf{ID}, R; r)$ and $c_3 = H(R, M, c_1, G(R) \oplus M)$ without asking $G(R)$, or (2)$\mathcal{B}$ submits a Decryption query $\langle \mathsf{ID}, c_1, c_2, H(R, M, c_1, c_2) \rangle$ without asking $H(R, M, c_1, c_2)$.

The case (1) and (2) happen with probability at most 2^{-l_2} and 2^{-l_3}, respectively, and therefore, we have that $\Pr[\mathsf{F}] \leq 1 - (1 - (1/2^{l_2}) - (1/2^{l_3}))^{q_D} \simeq q_D((1/2^{l_2}) + (1/2^{l_3}))$.

If $\mathcal{B}$ wins the $\mathsf{IND\text{-}ID\text{-}CCA}$ game, then $\mathcal{A}$ also win the $\mathsf{OW\text{-}ID\text{-}PCA}$ game. Therefore, $\Pr[\mathsf{SA}] \geq \Pr[\mathsf{SB}]$. Hence, we have that $Adv_{\mathcal{A}}(k) = \Pr[\mathsf{SA}] \geq \Pr[\mathsf{SB}] \simeq 2\epsilon(k) - q_D((1/2^{l_2}) + (1/2^{l_3}))$.

Finally, we estimate a running time of $\mathcal{A}$. Addition to the running time of $\mathcal{B}$, $\mathcal{A}$ has to answer the G and H queries. Thus $\mathcal{A}$'s running time is $t(k) + (q_H + q_G) \cdot O(1)$.

The Reduction Efficiency of REACT for IBE. As shown in Section 3.2, the reduction cost of the plain FOPKC is inefficient. Using our simple technique which we put ID to a part of the hash function's inputs, the reduction cost could significantly decreases. Unlike the plain FOPKC, the plain REACT already gives a tight reduction cost for IBE. We remark that this is mainly caused by the PC oracle, which implicitly handles the ID by its definition. The significant differences of reduction costs may indicate a separation between these two attack models: CPA and PCA.

5 Numerical Explanation

In this section, we evaluate how much better on the cost of security reduction the modified FOPKC is than the plain FOPKC by numerical explanation and show what this result means in the real world. As shown in Theorem 1, the plain FOPKC gives a polynomial time reduction and, looking at the coefficient of ϵ, the reduction seems tight, which means that the adversary's advantage against the underlying scheme and that of transformed scheme are close. Therefore, at a glance, merit of the modified FOPKC seems not considerable.

However, we notice that the other terms except for ϵ term have a significant influence on the reduction cost in the plain FOPKC. So, here, we compare the plain FO and our modified FOPKC by strictly estimating T' $(= t'/\epsilon')$ where T' is intuitively the average time for an adversary to succeed in the attack for the basic IBE scheme. We let Boneh and Franklin's scheme (BF-IBE) [7] be the underlying IBE scheme.

Parameter Setting. Let T' and T be t'/ϵ' and t/ϵ, respectively, where T' and T are the expected computational times to succeed in breaking the underlying IND-ID-CPA IBE scheme and the transformed IND-ID-CCA IBE scheme, respectively, assuming that a t'-time adversary can break the underlying IBE with advantage ϵ' and a t-time adversary can break the transformed IBE with advantage ϵ.

If the value T' is close to T, the reduction is said to be tight. On the other hand, if T' is much larger than T, the reduction is not tight and the adversary might not break the underlying IBE scheme in practical time. We derive the relation between T' and T of both the plain and the modified FOPKC.

In our estimation, we let $q_H = 2^{60}, q_D = 2^{40}$ and $\gamma = 2^{-160}$. We set that $l_1 - l_2 = 140$ which is the bit length of the random coin of both FOPKC conversions.

In the evaluation, we consider the case of $\epsilon = 2^{-10}$. Encrypting one message needs one pairing computation in BF-IBE scheme and this is the dominant part. The running time of fastest pairing algorithms in software and hardware are about 4.33 msec in software implementation (AthlonXP 2GHz)[2] and 0.85 msec in hardware implementation (FPGA 15Mhz)[15]. Thus, the running time of the encryption function τ is set to those values.

5.1 T' of the Plain and the Modified FOPKC

In the above setting, we evaluate T' of the plain FOPKC for BF-IBE:

$$T' \leq \frac{t + q_H q_D \tau}{(\epsilon + 1/2 - q_H/2^{(l_1 - l_2)})(1 - q_D\gamma) - 1/2} \simeq T + 2^{110} \times \tau.$$

The additional costs to break BF-IBE scheme $(T' - T)$ are 1.00×2^{102} sec in software implementation and 0.85×2^{100} sec in hardware implementation, respectively. Each of them needs too long time to break BF-IBE, of course, it is impossible to calculate in the real world.

In the above setting, we evaluate T' of the modified FOPKC for BF-IBE: $T' \leq T + 2^{110}\tau$. The additional costs to break BF-IBE $(T' - T)$ are 1.00×2^{62} in software implementation and 0.85×2^{60} in hardware implementation, respectively. In this case, the additional cost is much smaller than that in the plain FOPKC case.

The additional cost is almost for pairing calculations in encryptions and these operations are easily parallelized. Nowadays, it is not difficult to gather computing resources like millions order PCs or to produce a number of specialized IC chips. Thus, it can be said that this additional cost will be really feasible to compute in the near future even in the software implementation case.

Due to the above discussion, we see that if there exists an adversary who can break the modified FOPKC in a realistic time, then it is also possible to break the underlying IBE scheme in almost the same computational time. On the other hand, it is not clear whether the plain FOPKC provides the same level of security or not. Consequently, we can say that the modified FOPKC achieves *exact security* in a strict sense while the plain FOPKC does not.

References

1. N. Attrapadung, Y. Cui, D. Galindo, G. Hanaoka, I. Hasuo, H. Imai, K. Matsuura, P. Yang, and R. Zhang. Relations among notions of security for identity based encryption schemes. In *Latin American Theoretical Informatics (LATIN '06)*, volume 3887 of *LNCS*, pages 130–141. Springer, 2006.
2. P.S.L.M. Barreto. A note on effcient computation of cube roots in characteristic 3. Cryptology ePrint Archive, Report 2004/305, 2004. `http://eprint.iacr.org/2004/305`.
3. M. Bellare and P. Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416. Springer, 1996.
4. D. Boneh and X. Boyen. Efficient selective-ID identity based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT '04*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.
5. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology - CRYPTO '04*, volume 3152 of *LNCS*, pages 443–459. Springer, 2004.
6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology - CRYPTO '01*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

7. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. Full version of [6].
8. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology - EUROCRYPT '04*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.
9. C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proc. of the 8th IMA international conference on cryptography and coding*, volume 2260 of *LNCS*, pages 360–363. Springer, 2001.
10. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.
11. E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *Proc. of Public Key Cryptography 1999*, volume 1560 of *LNCS*, pages 53–68. Springer, 1999.
12. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.
13. E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. *IEICE Transactions Fundamentals*, E83-A(1):24–32, 2000. Full version of [11].
14. D. Galindo. Boneh-Franklin Identity Based Encryption Revisited. In *Proc. of 32nd ICALP*, volume 3580 of *LNCS*, pages 791–802. Springer, 2005.
15. T. Kerins, W.P. Marnane, E.M. Popovici, and P.S.L.M. Barreto. Efficient hardware for the Tate pairing calculation in characteristic three. In *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *LNCS*, pages 412–426. Springer, 2005. Presentation file is available from `http://islab.oregonstate.edu/ches/ches2005/presentations/`.
16. B. Libert and J.J. Quisquater. Identity based encryption without redundancy. In *Proc. of ACNS '05*, volume 3531 of *LNCS*, pages 285–300. Springer, 2005.
17. T. Okamoto and D. Pointcheval. REACT: rapid enhanced-security asymmetric cryptosystem transform. In *Topics in Cryptology – CT-RSA '01*, volume 2020 of *LNCS*, pages 159–174. Springer, 2001.
18. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO '84*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.
19. B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT '05*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.
20. P. Yang, T. Kitagawa, G. Hanaoka, R. Zhang, K. Matsuura, and H. Imai. Applying Fujisaki-Okamoto to identity-based encryption. In *Proc. of Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 3857 of *LNCS*, pages 183–192. Springer, 2006.

Tag-KEM from Set Partial Domain One-Way Permutations

Masayuki Abe[1], Yang Cui[2], Hideki Imai[3], and Kaoru Kurosawa[4]

[1] NTT Information Sharing Platform Laboratories, Japan
abe.masayuki@lab.ntt.co.jp
[2] University of Tokyo, Japan
cuiyang@imailab.iis.u-tokyo.ac.jp
[3] Chuo University, Japan
Research Center for Information Security (RCIS),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
h-imai@aist.or.jp
[4] Ibaraki University, Japan
kurosawa@mx.ibaraki.ac.jp

Abstract. Recently a framework called Tag-KEM/DEM was introduced to construct efficient hybrid encryption schemes. Although it is known that generic encode-then-encrypt construction of chosen ciphertext secure public-key encryption also applies to secure Tag-KEM construction and some known encoding method like OAEP can be used for this purpose, it is worth pursuing more efficient encoding method dedicated for Tag-KEM construction.

This paper proposes an encoding method that yields efficient Tag-KEM schemes when combined with set partial one-way functions such as RSA and Rabin's encryption scheme. We also present an efficient Tag-KEM which is CCA-secure under general factoring assumption rather than Blum factoring assumption.

Keywords: Tag-KEM, Hybrid Encryption, OAEP, SAEP.

1 Introduction

Hybrid encryption is a combination of public-key and symmetric-key encryption, which naturally compensates the shortcomings of sole use of each encryption scheme. The public-key encryption is used to convey one-time session key and the symmetric encryption is used to encrypt the actual long messages with the session key. While adaptive chosen-ciphertext security (CCA) [14] is the most desirable security for hybrid encryption, it has not been studied well what security is required for each building block to achieve the desirable security in the resulting hybrid encryption. The first general and formal treatment of hybrid encryption is done in [7, 16]. In their so-called KEM/DEM framework, public-key encryption is generalized to KEM (key encapsulation mechanism) and symmetric-key encryption is generalized to DEM (data encapsulation mechanism). They show a composition

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 360–370, 2006.
© Springer-Verlag Berlin Heidelberg 2006

theorem such that if both KEM and DEM is CCA-secure, then the resulting hybrid encryption is CCA-secure, too. While the KEM/DEM framework is useful in several aspects, it does not necessarily capture existing efficient schemes such as Kurosawa-Desmedt scheme [11], Fujisaki-Okamoto scheme [8], and etc.

In [1], another framework called Tag-KEM/DEM was introduced. It strengthens KEM to so-called Tag-KEM that takes an extra string as input and guarantees authenticity of the string just like a signature scheme does. On the other hand, the requirement to the DEM part is drastically relaxed in the framework so that it can only be one-time chosen-plaintext secure (CPA), which can be satisfied even by simple one-time pad. It is shown in [1] that some KEM can be transformed to Tag-KEM without increasing the ciphertext length. Furthermore, relatively simple CPA-secure DEMs often yield shorter ciphertexts than CCA-secure ones. Hence the resulting hybrid encryption schemes from the Tag-KEM/DEM framework tend to output shorter ciphertexts.

One approach for constructing a secure public-key encryption is to encode the plaintext and then put it into a trapdoor one-way permutation. There are many encoding schemes such as OAEP [4], OAEP+ [15], SAEP [5] and so on that achieve various security and efficiency features. They use hash functions, modeled as random oracles [3], to add redundancy for the plaintext so that validity can be tested when the ciphertext is decrypted. [1] shows that some of these encoding schemes can be used for constructing Tag-KEM just by including a tag to one of the hash functions. It demonstrates that Tag-KEM can be easily constructed with known technique but such constructions naturally inherit the cost of the underlying encoding schemes. This motivates us to design a new encoding scheme dedicated to Tag-KEM with better efficiency and security.

In this paper, we present a generic encoding method which yields efficient and CCA-secure Tag-KEM schemes when combined with set partial one-way permutation functions. It is known that set partial one-way can be implemented under RSA assumption [9] or Blum integer factoring assumption [6]. Hence our encoding method implies secure Tag-KEM schemes under these assumptions. We show concrete implementation under these assumptions. Since these concrete schemes have limitation to use Blum integers, we also show another scheme whose security can be reduced to general factoring assumption. Note that one can also obtain a general-factoring-based Tag-KEM by applying the generic conversion of [1] to the CCA-secure encryption of [13]. The resulting scheme, however, will be redundant and slower than our construction.

Compared to existing encoding methods such as OAEP or SAEP, our encoding method have advantage in the hash computation cost. Roughly speaking, the hash computation cost of our scheme is 1/3 as shown in Table 1. In particular, recent progress of attacks against hash functions seem to encourage the use of secure but more complicated and slower hash functions.

2 Definitions

We follow standard definition of public-key encryption and chosen ciphertext security. Some other relevant definitions follow [1] and are shown in the full version [2].

2.1 Tag-KEM

Tag-KEM is a set of algorithms (TKEM.Gen, TKEM.Key, TKEM.Enc, TKEM.Dec) such that

$(pk, sk) \leftarrow \mathbf{TKEM.Gen}(1^\lambda)$ A probabilistic algorithm that generates public-key pk and private-key sk. The public-key defines all relative spaces, i.e., spaces for tags and encapsulated keys denoted by $\mathcal{T}$ and $\mathcal{K}_K$.

$(\omega, dk) \leftarrow \mathbf{TKEM.Key}(pk)$ A probabilistic algorithm that outputs one-time key $dk \in \mathcal{K}_D$ and internal state information ω. $\mathcal{K}_D$ is the key-space of DEM.

$\psi \leftarrow \mathbf{TKEM.Enc}_{pk}(\omega, \tau)$ A probabilistic algorithm that encrypts dk (embedded in ω) into ψ along with τ, where τ is called a tag.

$dk \leftarrow \mathbf{TKEM.Dec}_{sk}(\psi, \tau)$ A decryption algorithm that recovers dk from ψ and τ. For soundness,$\mathsf{TKEM.Dec}_{sk}(\psi, \tau) = dk$ must hold for any sk, dk, ψ, and τ, associated by the above three functions. The algorithm can also output special symbol $\perp \notin \mathcal{K}_D$ to present abnormal termination.

Informally, the security of Tag-KEM requires the adversary to fail in distinguishing real session key embedded in a given ciphertext and a random string. Chosen ciphertext attack allows the adversary to access to the decryption oracle. Let $\mathcal{O}$ be the decryption oracle, $\mathsf{TKEM.Dec}_{sk}(\cdot, \cdot)$. Let A_T be a polynomial-time adversary. We define the following game:

[GAME.TKEM]

Step 1. $(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^\lambda)$
Step 2. $\upsilon_1 \leftarrow {A_T}^{\mathcal{O}}(pk)$
Step 3. $(\omega, dk_1) \leftarrow \mathsf{TKEM.Key}(pk)$, $dk_0 \leftarrow \mathcal{K}_D$, $\delta \leftarrow \{0, 1\}$.
Step 4. $(\tau, \upsilon_2) \leftarrow {A_T}^{\mathcal{O}}(\upsilon_1, dk_\delta)$
Step 5. $\psi \leftarrow \mathsf{TKEM.Enc}_{pk}(\omega, \tau)$
Step 6. $\tilde{\delta} \leftarrow {A_T}^{\mathcal{O}}(\upsilon_2, \psi)$

In Step 6 , A_T is restricted not to ask (ψ, τ) to decryption oracle $\mathcal{O}$. Variable υ_1, υ_2 are state information of the adversary. Variable dk_δ is set to either dk_0 or dk_1 according to the value of $\delta \in \{0, 1\}$. Such convention is used throughout the paper unless otherwise noted.

We say that A_T (ϵ_{tkem}, t)-breaks Tag-KEM if A_T runs in time t and

$$\left| \Pr[\tilde{\delta} = \delta] - \frac{1}{2} \right| \geq \epsilon_{tkem}.$$

We say that Tag-KEM is (ϵ_{tkem}, t)-secure if there is no A_T which can (ϵ_{tkem}, t)-break Tag-KEM. We may simply say that Tag-KEM is CCA-secure if the above holds for any negligible ϵ_{tkem} and any t bounded by polynomial in security parameter λ. In the random oracle model, we may include maximum number of oracle queries to the quantities. For instance $(\epsilon_{tkem}, t, q_H)$ denotes that the adversary makes up to q_H queries to random oracle H. The number of queries are supposedly bound by a polynomial of λ as well as other parameters.

Observe Tag-KEM has two encryption functions – $\mathsf{TKEM.Key}$ for generating session key and $\mathsf{TKEM.Enc}$ for encrypting the session key with a tag. This structure allows one to use the session key for DEM and then input the resulting ciphertext as a tag to $\mathsf{TKEM.Enc}$. [1] shows that such composition is secure. That is, if Tag-KEM is CCA-secure and DEM is one-time secure, then the resulting hybrid encryption scheme is CCA-secure. See [2] for formal treatment of DEM and relevant results from [1].

2.2 Set Partial Domain (SPD) One-Wayness

The notion of set partial domain one-wayness was introduced in [9].

Informally, set partial domain one-wayness states that, given a trapdoor permutation f and its output $y = f(r||x)$, it is hard to find a set of $S = \{r_1, \cdots, r_q\}$ such that $r \in S$.

Let $\mathcal{F}_n$ be a family of trapdoor permutations acting on strings in $\{0,1\}^n$. Let q be a parameter bounded by polynomial in n. We then define set partial one-wayness problem as follows. Given $f \leftarrow \mathcal{F}_n$ and $y = f(x)$ for random $x \in \{0,1\}^n$, output set $S = \{r_1, \ldots, r_q\} \subseteq \{0,1\}^s$ such that $r \in S$, where $x = r||z$ for some $z \in \{0,1\}^{n-s}$.

We say that a probabilistic algorithm $\mathcal{A}_{\mathsf{pow}}$ (q, t, ϵ)-breaks the set partial domain one-wayness problem on $\mathcal{F}$ with regard to s if $\mathcal{A}$ solves the above problem with running time at most t and probability ϵ. The probability is taken over the choice of f, x, and random coins of $\mathcal{A}$.

We say that $\mathcal{F}$ is (q, t, ϵ)-set partial domain one-way with regard to s if there is no probabilistic algorithm $\mathcal{A}_{\mathsf{pow}}$ which (q, t, ϵ)-breaks the set partial domain one-wayness problem. We may simply say that $\mathcal{F}$ is q-set partial domain one-way if any probabilistic poly-time machine $\mathcal{A}$ solves the problem only with negligible probability. It is proven in [9] that RSA is q-set partial domain one-way for any poly-bounded q.

3 Proposed Tag-KEM from SPD One-Wayness

Rough sketch of our scheme is as follows. We generate a random session-key from random coin r by applying a cryptographic hash function, which is considered as random oracle. Then, this random coin r and tag τ are encoded into $x = r||H(r, \tau)$ where H is another random oracle. Then x is encapsulated by applying a trapdoor permutation.

This section formally proves this simple and generic encode-then-encrypt construction yields Tag-KEM scheme when the trapdoor permutation is set partial one-way.

3.1 Proposed Tag-KEM

Let $\mathcal{F}_n$ be a family of trapdoor permutations acting on strings in $\{0,1\}^n$. Let $H : \{0,1\}^* \to \{0,1\}^{n-s}$ be a hash function.

TKEM.Gen(1^λ)**.** On the input of security parameter λ, generate a pair of (f_{pk}, f_{sk}), where f_{pk} is a public-key and f_{sk} is the trapdoor of $\mathcal{F}_n$.

TKEM.Key(pk)**.** Choose random seed $r \in \{0,1\}^s$ at random, and compute session key K by key derivation function H'

$$K = H'(r) \tag{1}$$

Then output (r, K).

TKEM.Enc$_{pk}(r, \tau)$**.** Given a tag τ, first compute

$$x = r||H(r, \tau) \tag{2}$$

where $||$ denotes concatenation, then output is equal to

$$\psi = f_{pk}(x) \tag{3}$$

TKEM.Dec$_{sk}(\psi, \tau)$**.** Compute

$$x = f_{sk}(\psi) \tag{4}$$

parse the above result as $x = r||h$, if $h = H(r, \tau)$, then output K which is given by eq.(1). Otherwise, output $\perp$.

3.2 Security

Informally, if the underlying PKE f is set partial domain one-way secure, then the proposed **TKEM** is CCA-secure. The following theorem formally states it.

Theorem 1. *Suppose that the underlying $\mathcal{F}_n$ is (q, t', ϵ')-set partial domain one-way with respect to s. Then for any adversary $\mathcal{A}$ that $(\epsilon, t, q_D, q_H, q_{H'})$-breaks the above Tag-KEM,*

$$\epsilon' \geq \epsilon - \frac{q_D}{2^n} - \frac{q_D}{2^{n-s}} \tag{5}$$

$$t' \leq t + O(q_D + q_H + q_{H'}) \tag{6}$$

where q_D, q_H, $q_{H'}$ denote the number of queries from $\mathcal{A}$ to the decryption oracle and random oracle H and H', respectively, and are bounded by $q_D + q_H + q_{H'} \leq q$. (in sense of section 2.2)

Proof. The outline of the proof follows the game-modifying method of [15]. We begin with a **TKEM** adversary $\mathcal{A}$ playing GAME.TKEM. Then we gradually modify the game so that we can use adversary $\mathcal{A}$ to solve a set partial one-way problem when it wins the game.

On the way of modifying the game, we use the following lemma to bound the probability transition. Proof of the lemma can be seen in [15].

Lemma 1. *[[7], Lemma 6.2] Let P, Q and F be events defined on some probability space, such that $\Pr[P \wedge \neg F] = \Pr[Q \wedge \neg F]$, then*

$$|\Pr[P] - \Pr[Q]| \leq \Pr[F]$$

GAME G_0: It is exactly the same as the original GAME.TKEM. Let E_0 be the event that $\tilde{b} = b$ in G_0. Similarly, let E_i for $i = 1, 2, 3, 4$ denote the same event in game G_i shown in the sequel.

GAME G_1: Let G_1 be modified from G_0, such that the following rejection rule is added: if the target ciphertext ψ^* was queried to decryption oracle before the invocation of the encryption oracle, the simulation is stopped immediately. Let F_1 be the above event. Since the same ciphertext happens with probability $1/2^n$, summing up to q_D decryption queries, it is easy to see that $\Pr[F_1] \leq q_D/2^n$. Note that G_1 and G_0 proceed identically unless F_1 occurs, by Lemma 1, we have

$$|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1] \leq \frac{q_D}{2^n}$$

GAME G_2: It is the same as G_1 except that an additional rejection rule is added to the decryption oracle. That is, the decryption oracle rejects (ψ, τ) if the corresponding (r, τ) has never been queried to H beforehand. Let F_2 be the event that the decryption oracle rejects an input in G_2 but the input is accepted in G_1. F_2 occurs only when $H(r, \tau)$ happens to match to the desired value defined by the input ciphertext. Since H is a random oracle, it happens only with probability $1/2^{n-s}$. Summing up to q_D decryption queries, we have $\Pr[F_2] \leq q_D/2^{n-s}$. Since G_2 and G_1 proceed identically unless F_2 occurs, we have the following from Lemma 1:

$$|\Pr[E_1] - \Pr[E_2]| \leq \Pr[F_2] \leq \frac{q_D}{2^{n-s}}$$

GAME G_3: Here, game G_2 is modified so that it aborts immediately on happening the event that H' oracle receives r^* or H oracle receives (r^*, τ^*). Let F_3 denote this event. Clearly, G_3 and G_2 proceed identically unless F_3 occurs. Hence from Lemma 1, we have

$$|\Pr[E_2] - \Pr[E_3]| \leq \Pr[F_3]$$

A crucial observation is that, if F_3 does not happen, $K^\dagger = H'(r^*)$ and $K^\dagger \neq H'(r^*)$ could happen equally likely because H' is a random oracle and $H'(r^*)$ is not defined. Therefore, the adversary has absolutely no advantage in distinguishing the cases and we can conclude $\Pr[E_3] = 1/2$.

GAME G_4: It is the same as G_3 except that the encryption oracle computes $(K^\dagger, \psi^*)$ at the beginning of the game. Since that computation is done based only on the choice of the random tape of the encryption oracle, it is independent from any other part of the game and hence G_4 and G_3 proceed identically. Accordingly, we have $\Pr[E_4] = \Pr[E_3] = 1/2$.

UPPER BOUND OF F_3: We give a upper bound of $Pr[F_3]$ by showing a reduction from adversary $\mathcal{A}$ that causes event F_3 to an algorithm, say $\mathcal{B}$ that breaks the set partial domain one-way assumption. On input $f \in \mathcal{F}$ and $y^* = f(r^*, h^*)$, $\mathcal{B}$ works as follows.

1. Set $\psi^* = y^*$ and choose $K^\dagger$ randomly.
2. Run $\mathcal{A}$ on input $f_{pk} = f$ and simulate random oracle H and H' and the decryption oracle by maintaining corresponding input/output lists. These lists, say H-list, H'-list, and D-list, are initially empty and have entry of the form (r, τ, h), (r, K), and (r, τ, y), respectively. (See below for notations.) For ease of description, we assume that all inputs to these oracles are fresh. Duplicated inputs are handled trivially by looking up the lists.
   H'-oracle: For every input r, simply select random K from the session-key domain. Then store (r, K) to the H'-list and return K.
   H-oracle: For every input (r, τ), select a random value $h \in \{0,1\}^{n-s}$, store (r, τ, h) to the H-list, and return h to $\mathcal{A}$. Additionally, compute $y = f_{pk}(r, h)$ and append entry (r, τ, y) to the D-list. (In this way, the H-list contains (r, τ, h) whenever (r, τ, y) is in the D-list.)
   Decryption oracle: For every query (y, τ), search the D-list for $(*, \tau, y)$. If there is no matching entry, reject the input due to the rejection rule settled in Game 1. Otherwise, take the first item in the matching entry as r. Then make a query r to the simulating H'-oracle, which works as described above, and return the given K.
3. Stop the simulation when $\mathcal{A}$ stops, or the running time of $\mathcal{A}$ comes to t, or any of the oracle queries achieves to the maximum number.
4. Pick up all the entry of r from the H-list and H'-list. Then output the obtained sequence $\{r_1, \ldots, r_t\}$.

First of all, observe that $\mathcal{B}$ simulates game G_4 perfectly unless F_3 occurs. (One important observation is that $K^* = H(r^*)$ is random because H is a random oracle. Therefore, $K^\dagger$ is random from a view point of $\mathcal{A}$ regardless of the value of b.) When F_3 happens, the return value from H (or H') is inconsistent and the simulation is no more correct. Nevertheless, since $\mathcal{B}$ cannot fetch the event F_3, the simulation continues until the adversary uses up one of its resources. However, this is no problem since r^* is already stored in either H-list or H'-list at the moment F_3 happens. And that r^* is included in the final output $\{r_1, \ldots, r_t\}$. Observe that the length t of the final answer does not exceed $q_D + q_H + q_{H'}$ by construction. Hence the list is a valid answer of the set partial domain one-way problem. Accordingly, we have $\Pr[F_3] \leq \epsilon'$.

Summing Up: By combing the series of games and the boundary observed by the simulation, we can compute the probability of $\mathcal{A}$ wins the GAME.TKEM.

$$
\begin{aligned}
\epsilon &\stackrel{\triangle}{=} |\Pr[E_0] - (1/2)| \\
&= |\Pr[E_0] - \Pr[E_1] + \Pr[E_1] - \Pr[E_2] \\
&\quad + \Pr[E_2] - \Pr[E_3]| + \Pr[E_3] - \Pr[E_4] \\
&\leq |\Pr[E_0] - \Pr[E_1]| + |\Pr[E_1] - \Pr[E_2]| \\
&\quad + |\Pr[E_2] - \Pr[E_3]| + |\Pr[E_3] - \Pr[E_4]| \\
&\leq \Pr[F_1] + \Pr[F_2] + \Pr[F_3] \\
&\leq q_D/2^n + q_D/2^{n-s} + \epsilon'
\end{aligned}
$$

Note that for emulating the oracles, the simulation should run in time $O(q_D + q_H + q_{H'})$ to keep and record the lists. Thus, total running time of simulation is bounded by $t + O(q_D + q_H + q_{H'})$. This completes the proof. □

4 Comparison

4.1 Previous Tag-KEM Based on SPD One-Wayness

A generic construction of CCA-secure Tag-KEM is shown in [1]. In this construction, TKEM.Key chooses random dk and TKEM.Enc computes

$$\psi = \mathsf{PKE.Enc}_{pk}(dk||\tilde{H}(\tau)), \tag{7}$$

where PKE.Enc is a CCA-secure encryption scheme and $\tilde{H}$ is a target collision-free hash function. TKEM.Dec computes $dk||\tau'$ from ψ. If $\tau' = \tilde{H}(\tau)$, it returns dk. Otherwise it returns $\perp$.

Proposition 1. *[[1], Theorem 2.] Given that PKE is CCA-secure, and a hash function $\tilde{H}$ is target collision-free, the Tag-KEM above is CCA-secure, such that,*

$$\epsilon_{tkem} \leq \epsilon_{pke} + \epsilon_{tch}$$

where ϵ_{tch} denotes the probability of finding a collision in $\tilde{H}$.

Furthermore, it is known that a CCA-secure encryption scheme is obtained from a family of set partial domain one-way permutations $\mathcal{F}$ in two ways, based on OAEP padding [4] and based on SAEP+ padding [5]. Accordingly, we can construct Tag-KEM schemes based on OAEP padding [4] and based on SAEP+ padding [5] if we assume that there exists a a family of set partial domain one-way permutations $\mathcal{F}$. These Tag-KEM schemes are illustrated in [2], Tag-KEM based in OAEP and Tag-KEM based on SAEP+.

In [1], another construction of Tag-KEM is shown which is based on OAEP+. OAEP+, however, requires 3 random oracles.

4.2 Efficiency

In the encryption of Tag-KEM, the efficiency of TKEM.Enc is more important than that of TKEM.Key because we can compute TKEM.Key in the preprocessing phase. Now our TKEM.Enc computes one hash function only as shown in eq.(2). On the other hand, each of OAEP padding and SAEP+ padding uses two random oracles. In addition, the generic construction shown in eq.(7) requires a target collision-free hash function $\tilde{H}$. Hence TKEM.Enc of these Tag-KEM must compute three hash functions. TKEM.Enc of Tag-KEM based on OAEP+ also must compute three hash functions.

In the decryption of Tag-KEM, the efficiency to check the validity of the ciphertext is important because no further computation is done if the ciphertext is rejected at this point. Now in our scheme, TKEM.Dec computes only one hash function for this purpose. On the other hand, TKEM.Dec of all the previous schemes must compute three hash functions.

Table 1. Number of hash computations

	OAEP	SAEP+	OAEP+	Proposed
TKEM.Enc	3	3	3	1
Validity check part of TKEM.Dec	3	3	3	1

Table 2. Number of pre-computation hash

	OAEP	SAEP+	OAEP+	Proposed
TKEM.Key	0	0	0	1

In the pre-processing phase, only one hash function needs to be computed, which takes a load off the on-line computation.

Using less hash computations is an important practical issue as the hash functions seem getting slower to circumvent the recent progress of attack methods.

5 Concrete Implementations

This section shows several concrete implementations of Tag-KEM based on the result of the previous section. We focus on introducing the features of each scheme and put detailed descriptions in the full version [2]. To show set partial domain one-wayness, the following proposition [6, Corollary 1] given by Coppersmith is used.

Proposition 2. *Given N and a monic polynomial $F(x)$ of degree d, one can find all roots x_0 such that $|x_0| \leq N^{1/d}$ and*

$$F(x_0) = 0 \bmod N$$

in polynomial time in $(\log_2 N, 2^d)$.

5.1 Based on RSA

It is known that RSA satisfies set partial domain one-wayness [9] under RSA assumption. Therefore, we can use RSA as $\mathcal{F}$ in the proposed Tag-KEM. The resulting Tag-KEM is CCA-secure under RSA assumption. However, this implementation inherits expensive security reduction cost as well as RSA-OAEP where the dominant factor is square of the number of hash queries.

5.2 Based on Rabin Function

Rabin encryption scheme is given by $f(x) = x^2 \bmod N$, where $N = pq$ with two primes p and q. Since it is not a permutation, we consider a variant such that $p = q = 3 \bmod 4$ and the message space is restricted to

$$\{x \mid (x/N) = 1, 0 < x < N/2\},$$

where $(\cdot/\cdot)$ denotes Jacobi symbol.

Let $n = |N|$ and $s = (n/2) + 1$. Then we can show that f satisfies set partial domain one-wayness with respect to s under Blum integer factoring assumption based on Proposition 2 by using the same as [5, Proof of Corollary 1]. Intuitively, this is explained as follows. First we can compute z from $y = (r||z)^2 \bmod N$ and $r \in \{0,1\}^s$ by using Proposition 2 because $z < N^{1/2}$. Next it is well known that computing a square root is equivalent to factoring. Therefore, if there exists an algorithm which breaks the set partial domain one-wayness, then we can construct an algorithm which can factor N.

Hence we can use this variant as $\mathcal{F}$ in the proposed Tag-KEM. The resulting Tag-KEM is CCA-secure under Blum integer factoring assumption from theorem 1.

5.3 Based on General Factoring

We finally show this efficient Tag-KEM which is CCA-secure under general factoring assumption. Kurosawa et al showed an encryption scheme [12] such that

$$f(x) = x + \frac{\alpha}{x} \mod N,$$

where $N = pq$ with two primes p and q, and

$$(\alpha/p) = (\alpha/q) = -1$$

where (α/p) is Legendre symbol.

Given message encoded in $\mathbb{Z}_N^*$: $m \in \mathbb{Z}_N^*$. Let ciphertext $C = (E, s, t)$ where,

$$E = m + \frac{\alpha}{m} \mod N$$

$$s = \begin{cases} 0 & \text{if } (m/N) = 1 \\ 1 & \text{if } (m/N) = -1 \end{cases} \qquad t = \begin{cases} 0 & \text{if } (\alpha/m \bmod N) > m \\ 1 & \text{if } (\alpha/m \bmod N) < m \end{cases}$$

Note that N is not a Blum integer. Also, note that we can obtain a quadratic equation on x from $y = f(x) \bmod N$. Then we can show that it satisfies the set partial domain one-wayness under general integer factoring assumption similarly to Sec.5.2. Hence we can use this variant as $\mathcal{F}$ in the proposed Tag-KEM. The resulting Tag-KEM is CCA-secure under general integer factoring assumption.

References

1. M. Abe, R. Gennaro, K. Kurosawa and V. Shoup. Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM. Eurocrypt 2005.
2. M. Abe, Y. Cui, H. Imai and K. Kurosawa. Tag-KEM from Set Partial Domain One-Way Permutations. (full version) See `http://eprint.iacr.org/`.
3. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communication Security*, pages 62–73. Association for Computing Machinery, 1993.

4. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption - How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92-111. Springer-Verlag, 1995.
5. D. Boneh. Simplified OAEP for the RSA and Rabin functions. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 275–291. Springer-Verlag, 2001.
6. D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerablities. *Journal of Cryptology*, vol. 10, pp. 233-260, 1997.
7. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
8. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.
9. E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA-OAEP Is Secure under the RSA Assumption. *J. Cryptology* 17(2): 81-104. 2004
10. R. Gennaro and V. Shoup. A note on an encryption scheme of Kurosawa and Desmedt. Technical Report 2004/194, IACR ePrint archive, 2004.
11. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In Matt Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer-Verlag, 2004.
12. K. Kurosawa, T. Itoh and M. Takeuchi. Public Key Cryptosystem using a Reciprocal Number with the Same Intractability as Factoring a Large Number. In *CRYPTOLOGIA*, XII, pp.225-233, 1988.
13. K. Kurosawa, W. Ogata, T. Matsuo, S. Makishima. IND-CCA Public Key Schemes Equivalent to Factoring n=pq. In *Public Key Cryptography 2001 (PKC'01)*, volume 1992 of *Lecture Notes in Computer Science*, pages 36–47. Springer-Verlag, 2001.
14. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992.
15. V. Shoup. OAEP reconsidered. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 239–259. Springer-Verlag, 2001.
16. V. Shoup. ISO 18033-2: An emerging standard for public-key encryption (committee draft). Available at `http://shoup.net/iso/`, June 3 2004.

An Extension to Bellare and Rogaway (1993) Model: Resetting Compromised Long-Term Keys*

Colin Boyd[1], Kim-Kwang Raymond Choo[1], and Anish Mathuria[2]

[1] Information Security Institute
Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia
[2] Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, Gujarat, India
boyd@isrc.qut.edu.au, raymond.choo.au@gmail.com,
anish_mathuria@da-iict.org

Abstract. A security proof in the Bellare–Rogaway model and the random oracle model is provided for a protocol closely based on one originally proposed by Boyd (1996), which enjoys some remarkable efficiency properties. The model is extended so that it can detect a known weakness of the protocol that cannot be captured in the original model. An alternative protocol is proposed, provably secure in the extended model and the random oracle model, and offering the same efficiency features as the original protocol. Moreover, our alternative protocol provides key confirmation and forward secrecy. It also allows session keys to be renewed in subsequent sessions without the server's further involvement even in the event that the long-term key or the earlier session key have been compromised.

1 Introduction

Protocols for key establishment are a foundational element in communications security. There has been an enormous amount of research effort expended in design and analysis of such protocols and yet there are still worthwhile contributions to be made even in the simple scenario of two users with an on-line server. For example, it is worthwhile to improve upon the performance cost associated with such protocols and ensure that the security goals can still be guaranteed.

Gong [9] has shown that protocols using timestamps require fewer messages and rounds than protocols using nonce-based challenge-response. Boyd [4] proposed a novel method of achieving key freshness which does not require both participants' nonces to be passed to the server, thus reducing the number of messages and rounds to the same as that required for timestamp-based protocols. However, a known weakness of Boyd's protocol class is that if a user's long-term key is compromised, then an attacker can masquerade as that user even after the compromised key is

* The full version of this paper appears in [5].

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 371–382, 2006.
© Springer-Verlag Berlin Heidelberg 2006

replaced with a new one. Moreover, Boyd's protocol class does not have a proof of security; its purported security is based on heuristic arguments. The main problem with the heuristic approach is that it does not provide a clear framework for defining a "secure" protocol and what constitutes an "attack". Since this approach does not account for all possible attacks, the security guarantees are limited and often insufficient. In contrast, the provable security paradigm for protocols provides a formal foundation for defining a "secure" protocol and allows rigorous proofs of security to be developed.

In this paper we prove the original protocol of Boyd secure in the widely accepted model of Bellare and Rogaway (hereafter referred to as the BR93 model) [3]. In the BR93 model, there exists a powerful adversary who can interact with all the participants, with an aim to learn some information about one session key. Therefore, one tries to prove the indistinguishability of the session key (from a random key) for the adversary. The BR93 model has been further revised several times by several other researchers. However, like many other users of these models, we find that they are insufficiently rich to capture all reasonable actions of the adversary. In a practical system we may expect that once the compromise of a user has been detected, that user will be reset with a new long-term key and then allowed to continue working. In the type of protocols we are concerned with this scenario will allow the adversary to masquerade as that user. However, since there is no notion of resetting in the BR93 model there is no way to observe such a possibility. Therefore we extend the model to allow more capabilities for the adversary.

We then propose an alternative protocol, equally efficient in terms of messages and rounds, that provides protection against the compromise of long-term keys without taking recourse to revocation lists.

Contributions of Paper. The contributions of this paper are three-fold. (1) A revised protocol of Boyd [4] is proven secure in the BR93 model and the random oracle model (also known as the ideal hash model). (2) The BR93 model is extended to allow more realistic adversary capabilities, under which the proven secure protocol of Boyd becomes insecure. Protocols proven secure in the extended model will also be secure in the original model. (3) An alternative protocol that is efficient in both messages and rounds is then shown to be secure in the extended BR93 model and the random oracle model. It provides key confirmation and forward secrecy[1] and allows session keys to be renewed in subsequent sessions without the server's further involvement (i.e., re-authentication) even in the event that the long-term key or the earlier session key have been compromised. We remark that there are very few server-based protocols that achieve forward secrecy and allow re-authentication in the event that the long-term key or the earlier session key have been compromised.

[1] 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*.

Organization of Paper. Section 2 reviews the BR93 model and the mathematical preliminaries. Section 3 describes a protocol closely based on one originally proposed by Boyd [4] and provides a proof of its security in the BR93 model. Section 4 describes the limitation of the proof for the original protocol and extends the model so that there is capability to reset long-term keys. Section 5 describes an alternative protocol and provides a proof of its security in the extended model. A comparative summary is presented in Section 6. An extension to this alternative protocol allows session keys to be renewed in subsequent sessions without the server's further involvement even in the event that the long-term key or the earlier session key have been compromised is also described in this section. Section 7 presents the conclusions.

2 Provable Security Paradigm for Protocols

Bellare and Rogaway provide 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) in their 1993 paper [3] where they provide mathematical proofs for two-party entity authentication protocols. In the model, there exist a powerful adversary who can interact with all the participants, with an aim to learn some information about one session key. Therefore, one tries to prove the indistinguishability of the session key (from a random key) for the adversary.

2.1 The Adversarial Model

Informally the adversary, $\mathcal{A}$, is allowed to fully control the communication network by injecting, modifying, blocking, and deleting any messages at will. $\mathcal{A}$ can also request for any session keys adaptively. The adversary interacts 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. An oracle, Π_U^s, represents the actions of principal U in the protocol run indexed by integer s. Formally, $\mathcal{A}$ can adaptively query the following oracles, as follows:

$\mathsf{Send}(U, s, m)$. This query allows $\mathcal{A}$ to make U runs the protocol normally. Π_U^s will return to $\mathcal{A}$ the same next message that an honest principal, U, would if sent message m according to the conversation so far. If Π_U^s accepts the session key or halts this is included in the response. $\mathcal{A}$ can also use this query to initiate a new protocol instance by sending an empty message m.

$\mathsf{Reveal}(U, s)$. This query models $\mathcal{A}$'s ability to find session keys. If a session key, K_s, has previously been accepted by Π_U^s, then it is returned to $\mathcal{A}$. An oracle can only accept a key once. An oracle is called *unfresh* if it has been the object of a Reveal query.

$\mathsf{Corrupt}(U)$. This query returns the oracle's long-term secret key. A principal is called *corrupted* if it has been the object of a Corrupt query. Note that this query does not return the session key since session keys can be learnt by the Reveal query or the entire internal state.

$\mathsf{Test}(U, s)$. Once Π_U^s has accepted a session key, K_s, $\mathcal{A}$ 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 $\mathcal{A}$.

2.2 Definition of Security

Definition of security in the BR93 model depends on the notion of the *partner* oracles to any oracle being tested. The way of defining partner oracles has varied in different papers using the model. Following recent trends, we define SID_U^s as the concatenation of all messages that oracle Π_U^s has sent and received.

Definition 1. *Two oracles are* partnered *if (1) they have accepted a session key with the same session identifier (SID), (2) each believes that the other is its partner, and (3) they agree on the initiator of the protocol.*

Definition 2 describes the freshness definition.

Definition 2. *An oracle Π_U^s is* fresh *at the end of its execution if (1) Π_U^s has accepted with partner Π_V^t (if such a partner exists), (2) Π_U^s and Π_V^t are unopened, and (3) principals U and V are uncorrupted.*

The security of the protocol is defined by the following game, $\mathcal{G}$, played between the adversary and an infinite collection of user oracles Π_U^s for $U \in \{U_1, \ldots, U_Q\}$ and $s \in \mathbb{N}$ and server oracles Π_S^s. Firstly, long-lived keys are assigned to each user by running the key distribution algorithm $\mathcal{K}_k$ on input of the security parameter k. Then, the adversary, $\mathcal{A}(1^k)$, is run. $\mathcal{A}$ will interact with the oracles through the queries defined above. At some stage during the execution a Test query is performed by the adversary to a fresh user oracle. Eventually the adversary outputs a bit b' and terminates. Success of the adversary, $\mathcal{A}$, in this game is measured in terms of its *advantage* in distinguishing the session key of the Test query from a random key, i.e., its advantage in outputting $b' = b$. This advantage must be measured in terms of the security parameter k. If we define $\mathsf{success}$ to be the event that $\mathcal{A}$ guesses correctly whether $b = 0$ or $b = 1$, then

$$\mathsf{Adv}^{\mathcal{A}}(k) = |2 \cdot \Pr[\mathsf{success}] - 1|.$$

To define validity of a protocol, we use the concept of a *benign adversary* as an adversary that faithfully relays flows between participants [3].

Definition 3. *A protocol P is a* secure key establishment protocol *if the following two properties are satisfied:*

Validity. *In the presence of a benign adversary partner oracles conclude with the same key except for a negligible probability.*

Indistinguishability. *For every probabilistic polynomial-time adversary, $\mathcal{A}$, the function $\mathsf{Adv}^{\mathcal{A}}(k)$ is negligible.*

Security of a protocol is proved by finding a reduction to some well known computational problem whose intractability is assumed (i.e., in this paper, the Computational Diffie-Hellman (CDH) problem). In addition, we require the notion of an authenticated encryption scheme, which forms the basis of our proof for Protocol 2 described in Section 5.

2.3 The Computational Diffie-Hellman Assumption

Let $\mathbb{G} \in \mathbb{Z}_p^*$ be a cyclic group of prime order q and g is assumed to be a generator of $\mathbb{G}$, where $\mathbb{G}$ is of prime order. The security parameters, p and q, are defined as the fixed form $q|p-1$ and $ord(g) = q$.

Computational Diffie-Hellman (CDH) Problem. Given an instance, (g, g^x, g^y), output g^{xy}.

A Computational Diffie-Hellman (CDH) attacker, $\mathcal{F}_{CDH}$, in a finite cyclic group $\mathbb{G}$ of prime order q with g as a generator, is a probabilistic machine, $\triangle$, running in time t such that the success probability of $\mathcal{F}_{CDH}$ when given random elements, $g^{N_1} \in \mathbb{G}$ and $g^{N_2} \in \mathbb{G}$ to output $g^{N_1 N_2} \in \mathbb{G}$, is less than ϵ, where the probability is over the random choice of N_1 and N_2 in $\mathbb{Z}_q^*$. In other words, the CDH assumption states that the success probability of $\mathcal{F}_{CDH}$ for any $\frac{t}{\epsilon}$ is not too large.

2.4 Secure Authenticated Encryption Schemes

We now define the authenticated encryption scheme that will be employed in the protocol that we shall prove secure in Section 3.

Let k denote the security parameter. A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms, namely: the *key generation* algorithm $\mathcal{K}$, the *encryption* algorithm $\mathcal{E}$, and the *decryption* algorithm $\mathcal{D}$ as described below.

- $\mathcal{K}$ is a probabilistic algorithm which, on input 1^k, outputs a key K.
- $\mathcal{E}$ is a probabilistic algorithm which takes a key K and a message M drawn from a message space $\mathcal{M}$ associated to K and returns a ciphertext C. This is denoted by $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M)$.
- $\mathcal{D}$ is a deterministic algorithm which takes a key K and a ciphertext C and returns the corresponding plaintext M or the symbol $\perp$ which indicates an illegal ciphertext. This is denoted as $x \leftarrow \mathcal{D}_K(C)$. We require that $\mathcal{D}_K(\mathcal{E}_K(M)) = M$ for every $K \leftarrow \mathcal{K}(1^k)$.

For security we use the definitions of Bellare & Namprempre [1]. We require that the symmetric encryption scheme provides confidentiality in the sense of indistinguishability under chosen plaintext attacks (*IND-CPA security*) and provides integrity in the sense of preserving integrity of plaintexts (*INT-PTXT security*). We note that each of these is the weakest of the properties defined by Bellare and Namprempre and are provided by either encrypt-then-MAC or by MAC-then-encrypt constructions. Therefore there are many practical ways of implementing

our protocol which can reasonably be expected to satisfy these assumptions. We now define these concepts more precisely.

For any efficient (probabilistic polynomial time) adversary, $\mathcal{X}$, the confidentiality security is defined in terms of the following game, which we call $\mathcal{G}_1$.

1. The challenger chooses a key $K \leftarrow \mathcal{K}(1^k)$.
2. Given access to the encryption oracle, the adversary outputs two messages of equal length $M_0, M_1 \in \mathcal{M}$ of her choice.
3. The challenger computes $C_b \stackrel{R}{\leftarrow} \mathcal{E}_K(M_b)$ where $b \stackrel{R}{\leftarrow} \{0,1\}$. The bit b is kept secret from the adversary.
4. The adversary is then given C_b and has to output a guess b' for b.

We define the advantage of the adversary, $\mathcal{X}$, playing the above game as

$$\mathsf{Adv}_{\mathcal{X}}^{\mathrm{ind-cpa}}(k) = |2 \cdot \Pr[b' = b] - 1|.$$

Definition 4. *The encryption scheme $\mathcal{SE}$ is IND-CPA secure if the advantage of all efficient adversaries playing game $\mathcal{G}_1$ is negligible.*

For any efficient adversary, $\mathcal{F}$, the integrity security is defined in terms of the following game, which we call $\mathcal{G}_2$.

1. Choose a key $K \leftarrow \mathcal{K}(1^k)$.
2. The adversary, $\mathcal{F}$ is given access to the encryption oracle and also a *verification oracle* which on input a ciphertext C outputs 0 if $\mathcal{D}_K(C) = \perp$ and outputs 1 if C is a legitimate ciphertext.
3. The adversary wins if it can find a legitimate ciphertext C^* such that the plaintext $M = \mathcal{D}_K(C^*)$ was never used as a query to the encryption oracle. In this case we say the event $\mathsf{forgery}$ has occurred.

We define the advantage of the adversary playing the above game as

$$\mathsf{Adv}_{\mathcal{F}}^{\mathrm{int-ptxt}}(k) = |2 \cdot \Pr[\mathsf{forgery}] - 1|.$$

Definition 5. *The encryption scheme $\mathcal{SE}$ is INT-PTXT secure if the advantage of all efficient adversaries playing game $\mathcal{G}_2$ is negligible.*

3 A Provably-Secure Revised Protocol of Boyd

Protocol 1 is a server-based protocol in which users A and B as well as the server S contribute to the key value. All parameter choices depend on a security parameter k. In Protocol 1, the following notations are used: $\{m\}_K$ denotes an authenticated encryption of some message m under symmetric key K; S denotes a server who shares long-term symmetric keys K_{AS} and K_{BS} with A and B, respectively; N_A, N_B, and K_S denote nonces generated by A, B and S, respectively; and $\mathcal{H}$ is modelled as a random oracle. The session key obtained by A and B at the end of the protocol execution is denoted as K_{AB}.

Protocol 1 is very similar to that proposed by Boyd [4]. Differences are as follows.

A	S	B
$N_A \in_R \{0,1\}^k$	$\xrightarrow{A, B, N_A}$ $K_S \in_R \{0,1\}^k$	$\xrightarrow{\{A,B,K_S\}_{K_{AS}}, \{A,B,K_S\}_{K_{BS}}, N_A}$
		$N_B \in_R \{0,1\}^k$
Decrypt $\{A, B, K_S\}_{K_{AS}}$	$\xleftarrow{\{A,B,K_S\}_{K_{AS}}, N_B}$	Decrypt $\{A, B, K_S\}_{K_{BS}}$
$SID_A = N_A \parallel N_B$		$SID_B = N_A \parallel N_B$
$K_{AB} = \mathcal{H}(K_S \parallel SID_A)$		$K_{AB} = \mathcal{H}(K_S \parallel SID_B)$
Status: ACCEPTED		Status: ACCEPTED

Protocol 1. A revised key agreement protocol of Boyd

1. In the earlier protocol of Boyd, the session key is determined by a MAC function so that the session key is $K_{AB} = MAC_{K_S}(N_A, N_B)$.
2. There is no partnering mechanism (e.g., session identifiers) specified in the earlier protocol of Boyd. Message exchanges in the real world are seldom conducted over secure channels. Therefore, it is realistic to assume that any adversary is able to modify messages at will, which is the case in the Bellare–Rogaway style models. As Goldreich and Lindell [7, Section 1.3] have pointed out, such an adversary capability means that the adversary is able to conduct concurrent executions of the protocol (one with each party). Therefore, without such partnering mechanism, communicating parties will be unable to uniquely distinguish messages from different sessions. Hence, in Protocol 1, we define partnership using the notion of session identifiers, SID[2].
3. The key confirmation messages have been removed, which consist of a handshake using the shared secret. These can easily be added in a standard way [2]. The session key itself must not be used to authenticate the key confirmation messages, otherwise the adversary can use them to easily distinguish the session key.

In the full version of this paper [5], we show that if the authenticated encryption algorithm used in Protocol 1 is secure, then Protocol 1 is also secure. We then arrive at Theorem 1.

Theorem 1. *Let $\mathcal{A}$ be any polynomial time adversary against the security of the protocol and $\mathcal{H}$ is modelled as a random oracle. Then there is an integrity adversary, $\mathcal{F}$, and a confidentiality adversary, $\mathcal{X}$ against the encrypted authentication algorithm such that*

$$\Pr(\mathsf{success}_{\mathcal{A}}) \leq Q \cdot \Pr(\mathsf{success}_{\mathcal{F}}) + Q^2 \cdot Q_S \cdot Q_H \cdot \Pr(\mathsf{success}_{\mathcal{X}}).$$

[2] The security proof of Protocol 1 does not hinge on the difficulty of predicting a valid session identifier. In fact, we may assume that session identifiers are made publicly available when the status of the principal becomes "ACCEPTED".

4 An Extension to the BR93 Model

Despite Protocol 1 being proven secure, it has a significant weakness in a realistic setting (similar to the weakness acknowledged by Boyd in his protocol [4]). It is inevitable that from time to time long-term keys of users will be compromised, e.g., theft of a device containing the key. It seems natural that in such a case the user should be re-issued with a new long-term private key and then allowed to continue using the protocol. For many server-based protocols this procedure will not influence the protocol security. However, for Protocol 1 this is not the case. It is easy to see that an adversary who obtains a long-term key of a user can continue to use it to masquerade as that user even after a new long-term key has been issued. The reason that this attack is possible even though we have proven the protocol secure, is that there is no notion of replacing a long-term key in the BR93 model: once a party has been corrupted it must remain so. In other words, once a party, say U_1, is corrupted and its long-term key revealed to the adversary, $\mathcal{A}$, U_1 is no longer considered fresh in the sense of Definition 2.

One of the motivations for this work is to remove a known weakness of the protocol of Boyd [4] under the effect of a compromise of a long-term key. That is, even if the adversary, $\mathcal{A}$, has corrupted some party, say U_1, $\mathcal{A}$ should not be able to impersonate U_1 using the compromised long-term key (of U_1) after a new long-term key has been issued to U_1. In order to take into account this sort of attack we add a new query called **Reset** to the list of actions that an adversary is allowed to perform and adjust the definition of freshness.

***Reset** Query.* The $\mathsf{Reset}(U_i, K_{New})$ query captures the notion of replacement for a compromised long-term key of principal U_i with a new randomly distributed key, K_{New}. When a corrupted U_i is being asked such a Reset query,

- player U_i is re-considered fresh in the sense of Definition 2,
- any oracle(s) $\Pi_{U_i}^1, \ldots, \Pi_{U_i}^{\delta-1}$ that were activated before the Reset query are unfresh in the sense of Definition 2, and
- subsequent oracles $\Pi_{U_i}^{\delta}, \Pi_{U_i}^{\delta+1}, \ldots$ are considered fresh in the sense of Definition 2 (unless U_1 is corrupted again).

An adversary, $\mathcal{A}$ who has access to this new query can always defeat Protocol 1 as follows.

1. $\mathcal{A}$ uses Send queries to run the protocol between A and B.
2. Then $\mathcal{A}$ issues a $\mathsf{Corrupt}(A)$ query to obtain the long-term key of A. This enables $\mathcal{A}$ to decrypt the ticket $\{A, B, K_S\}_{K_{AS}}$ sent to A during a previous protocol run with B, and hence obtain the key K_S contained in it.
3. $\mathcal{A}$ now resets A and masquerades as S, replaying the ticket originally sent to B together with any random value for N_A. This activates a fresh oracle Π_B^s, that will choose a nonce N_B and accept the session key $\mathcal{H}(K_S \parallel N_A \parallel N_B)$.
4. Consequently, $\mathcal{A}$ knows the value of this accepted key, in violation of Definition 3.

In order to avoid the problem, one method is to introduce a validity period for tickets and to issue a blacklist for tickets that have been compromised. This is the method suggested by Crispo, Popescu, and Tanenbaum [6] whereby they show that a large number of users can be accommodated in a practical system. It is easily checked that this prevents the above attack, since revoked tickets cannot be replayed by the adversary. However, such an approach entails a considerable infrastructure (not unlike a public key infrastructure) and might not scale well to a more realistic environment with a large number of participating entitites.

5 An Efficient and Provably-Secure Protocol in the Extended Model

Protocol 2 describes our proposed key agreement protocol. In Protocol 2, $\mathcal{H}_0$ and $\mathcal{H}_1$ are modelled as random oracles, $[\cdot]_{MK}$ denotes the computation of some MAC digest using MAC key, MK, $\{\cdot\}_{K_{US}}$ denotes the encryption of some message using encryption key, K_{US}, that is being shared by some user and the server, and $||$ denotes the concatenation of messages. We assume that $\mathbb{G}$, q, g, $\mathcal{H}_0$, $\mathcal{H}_1$ are fixed in advance and known to the entire network, and that each party P_i has a long-term symmetric key, K_{P_iS}, shared with the server, S.

A	S	B
$N_A \in_R \{0,1\}^k$	$\{A, B, g^{N_A}\}_{K_{AS}} \xrightarrow{\quad} \quad \{A, B, g^{N_A}\}_{K_{BS}} \xrightarrow{\quad}$	
		$N_B \in_R \{0,1\}^k; SID_B = g^{N_A}\|\|g^{N_B}$
		$MK_{AB} = \mathcal{H}_1(A\|\|B\|\|SID_B\|\|(g^{N_A})^{N_B})$
		$K_{AB} = \mathcal{H}_0(A\|\|B\|\|SID_B\|\|(g^{N_A})^{N_B})$
$SID_A = g^{N_A}\|\|g^{N_B}$	$g^{N_B}, [\text{“1”}, B, A, SID_B]_{MK_{AB}} \xleftarrow{\quad}$	Delete N_B
$MK_{AB} = \mathcal{H}_1(A\|\|B\|\|SID_A\|\|(g^{N_B})^{N_A})$		
Verify received MAC digest, $[\text{“1”}, B, A, SID_B]_{MK_{AB}}$		
$K_{AB} = \mathcal{H}_0(A\|\|B\|\|SID_A\|\|(g^{N_B})^{N_A})$		
Delete N_A	$\xrightarrow{\quad} [\text{“2”}, A, B, SID_A]_{MK_{AB}}$	Verify $[\text{“2”}, A, B, SID_A]_{MK_{AB}}$
Status: ACCEPTED		Status: ACCEPTED

Protocol 2. A new key agreement protocol with key confirmation and forward secrecy

Informally, Protocol 2 removes the known weakness of Protocol 1, as described below.

1. Upon completion of an execution of Protocol 2, A and B have accepted session keys of the same value, $K_{AB} = \mathcal{H}_0(A||B||g^{N_A}||g^{N_B}||g^{N_B N_A})$.
2. Suppose the adversary, $\mathcal{A}$, compromises the long-term key of A, K_{AS}. With knowledge of K_{AS}, $\mathcal{A}$ can decrypt $\{A, B, g^{N_A}\}_{K_{AS}}$ and learn g^{N_A}. $\mathcal{A}$ also knows

g^{N_B} from observing the Protocol 2's execution. However, finding $g^{N_B N_A}$ is equivalent to solving the CDH problem (recall that N_A has been deleted from the internal state of A upon completion of the execution of Protocol 2). Moreover, this implies that Protocol 2 provides *forward secrecy* since the knowledge of the compromised long-term keys, K_{AS} or K_{BS}, does not allow the adversary to find the session key, $K_{AB} = \mathcal{H}_0(A||B||g^{N_A}||g^{N_B}||g^{N_B N_A})$.

Theorem 2. *Assuming the Computational Diffie-Hellman (CDH) assumption is satisfied in $\mathbb{G}$, Protocol 2 is a secure key agreement protocol providing key confirmation and forward secrecy when $\mathcal{H}_0$ and $\mathcal{H}_1$ are modeled as random oracles and if the underlying message authentication scheme and encryption scheme are secure in the sense of existential unforgeability under adaptive chosen-message attack and indistinguishable under chosen-plaintext attack respectively.*

The proof for Theorem 2 appears in [5].

6 Comparative Security and Efficiency

Similar to the work of Gong [9] and Boyd [4], our motivation is to design protocols efficient in both messages and rounds. Therefore, we present a comparative summary of Protocols 1 and 2 with other similar server-based key establishment protocols of Gong [8,9] as described in Table 1. In particular, we compare Protocols 1 and 2 with the protocol classes defined by Gong where both users contribute to the session key.

In terms of both messages and rounds, we observe that

- Protocol 1 is as efficient as that obtained by Gong [9] for server-based protocols with similar goals using timestamps.
- Protocol 2, which provides key confirmation, breaks Gong's lower bound since an extra round is required for providing key confirmation in the first three protocols described in described in Table 1.

Moreover, Protocol 2 removes the known weakness of Protocol 1 under the effect of a compromise of a long-term key as described in Section 5 at the expense of computational overhead (i.e., Protocol 2 is more computational expensive due to the use of Diffie–Hellman exponentation).

We also remark that another attractive feature of Protocol 2 is the extension which allows session keys to be renewed in subsequent sessions without the server's further involvement. The extension to Protocol 2 that allows the session key to be renewed is described in Protocol 3. This entails A and B exchanging new nonces N'_A and N'_B and computing the new session key as $K'_{AB} = \mathcal{H}_1(A||B||S||N'_A||N'_B||g^{N_A N_B}) = K'_{BA}$.

7 Conclusions

We proved the security of another protocol example, revised protocol of Boyd [4] – Protocol 1, in the BR93 model. Although Protocol 1 is known to be insecure

Table 1. A comparative summary

	Protocols	Messages	Security proof?
The following three protocols do not provide key confirmation (KC). However, key confirmation can be provided at the cost of an extra message.			
1.	Protocol 1	3 (+1 for KC)	Proven secure in the BR93 model.
2.	Timestamp-based protocol [9]	4 (+1 for KC)	No.
3.	Nonce-based protocol [9]	5 (+1 for KC)	No.
The following three protocols provide key confirmation.			
4.	Alternative protocol using uncertified keys [9]	5	No.
5.	Hybrid protocol [8]	5	No.
6.	Protocol 2	4	Proven secure in the extended BR93 model. Protocols proven secure in the extended BR93 model will also be secure in the BR93 model. Moreover, Protocol 2 provides both key confirmation and forward secrecy.

A		B
$N'_A \in_R \{0,1\}^k$	$\xrightarrow{A, N'_A}$ $\xleftarrow{B, N'_B}$	$N'_B \in_R \{0,1\}^k$
	$SID_{A'} = (N'_A \| N'_B) = SID_{B'}$	
$K'_{AB} = \mathcal{H}_1(A\|B\|S\|SID_{A'}\|g^{N_A N_B}) = \mathcal{H}_1(A\|B\|S\|SID_{B'}\|g^{N_A N_B}) = K'_{BA}$		

Protocol 3. An extension to Protocol 2

under reasonable assumptions, this does not show up in the original BR93 model because there is no capability for the adversary to reset corrupted principals. We then extended the BR93 model so that it allows more realistic adversary capabilities, which allows us to detect a known weakness of Protocol 1 that cannot be captured in the original (BR93) model. We then presented another protocol (i.e., Protocol 2) that is efficient in both messages and rounds, and then proved Protocol 2 secure in the extended BR93 model and the random oracle model.

Future Work. This work allows us to detect a known weakness of the Boyd key agreement protocol [4] that cannot be captured in the original BR93 model. It would be interesting to know what other (symmetric-key) protocols may also have this property. Another possible extension is to investigate and propose a

modular proof approach with a formal statement of security that allows server-based three-party key establishment protocols like those introduced in Table 1 to renew session key(s) in subsequent sessions without the server's further involvement, even in the event that the long-term key or the earlier session key are compromised.

References

1. M. Bellare and C. Namprempre. Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm. In *ASIACRYPT 2000*, volume 1976/2000 of *LNCS*, pages 531–545. Springer-Verlag, 2000.
2. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *EUROCRYPT 2000*, volume 1807/2000 of *LNCS*, pages 139 – 155. Springer-Verlag, 2000.
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO 1993*, volume 773/1993 of *LNCS*, pages 110–125. Springer-Verlag, 1993.
4. C. Boyd. A Class of Flexible and Efficient Key Management Protocols. In *CSFW 1996*, pages 2–8. IEEE Computer Society Press, 1996.
5. C. Boyd, K.-K. R. Choo, and A. Mathuria. An Extension to Bellare and Rogaway (1993) Model: Resetting Compromised Long-Term Keys (Full version available from `http://sky.fit.qut.edu.au/~boydc/papers/`). In *ACISP 2006*, LNCS. Springer-Verlag, 2006.
6. B. Crispo, B. C. Popescu, and A. S. Tanenbaum. Symmetric Key Authentication Services Revisited. In *ACISP 2004*, volume 3108/2004 of *LNCS*, pages 248–261. Springer-Verlag, 2004.
7. O. Goldreich and Y. Lindell. Session-Key Generation using Human Passwords Only (Updated Version available from `http://eprint.iacr.org/2000/057/`). In *CRYPTO 2001*, volume 2139/2001 of *LNCS*, pages 408–432. Springer-Verlag, 2001.
8. L. Gong. Using One-Way Functions for Authentication. *ACM SIGCOMM Computer Communications Review*, 8(11):8–11, 1989.
9. L. Gong. Lower Bounds on Messages and Rounds for Network Authentication Protocols. In *ACM CCS 1993*, pages 26–37. ACM Press, 1993.

Graphical Representation of Authorization Policies for Weighted Credentials

Isaac Agudo, Javier Lopez, and Jose A. Montenegro

Computer Science Department, E.T.S. Ingenieria Informatica
University of Malaga, Spain
{isaac, jlm, monte}@lcc.uma.es

Abstract. This paper elaborates on a solution to represent authorization and delegation in a graphical way, allowing users to better interpret delegation relationships. We make use of Weighted Trust Graph (WTG) as an instrument to represent delegation and authorization, extending it to cope with more complicated concepts, and providing a graphical representation of the level of confidence that exists between two entities regarding a resource or attribute. We represent the level of confidence for each pair of entities as a point in an axis diagram, as a set of points, or as a set of triangular regions depending on the accuracy we need. Then, we use the same diagram to represent the set of acceptable confidence level, that we call authorization policy set. In this way, a single diagram can be used to decide about authorization, thus providing a powerful tool for systems in which interaction of users is needed.

1 Introduction

Logic programming offers a nice mechanism to represent authorization and delegation statements and decisions (see [5, 6, 2] for a list of examples) Statements are represented as predicates and decisions are based on formulae verification. There are many logical solutions for formulae verification and it is not difficult to implement them. However, one disadvantage of logical programming is that it is not easy to understand and has an obscure transcription. Moreover, the syntaxis of the different solutions are not homogeneous and, as a consequence, the learning process of the syntax can be quite hard. When trying to use logic directly, one has to deal with too many details that might be avoided if one makes use of a more user-oriented solution. In some sense, logic could be useful in a second stage, that is, not in the specification phase but in the analysis or decision-making phase.

On the other hand, there are solutions that, though less powerful, are more expressive and easier to understand. One of them is to use graphs, and in particular, *directed graphs*. Proposals that make use of directed graphs to model authorization and delegation statements use to map each predicate to a directed arc in a graph. Arcs go from the issuer of the authorization or delegation statement to the subject who is granted privileges. Thus, the graph includes as many different arcs as different authorization/delegation statements are considered.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 383–394, 2006.
© Springer-Verlag Berlin Heidelberg 2006

The result is a tree where the root usually is the owner of the resource we are reasoning about. That tree helps to understand the relations among entities in the system in a graphical way.

Varadharajan et al. have proposed two solutions to represent authorization and delegation using directed graphs. In [3] they presented a basic approach that support graphical representation of positive authorization, negative authorization and delegation. This solution follows the *predecessor-take-precedence* policy to resolve conflicts between positive and negative authorization. In [4] they presented a more elaborated proposal that makes use of integer numbers to assign a certainty level to each credential.

Weighted Trust Graphs (WTG), presented in [1] aims to generalize this proposal, defining it in a more flexible way. In fact, that proposal is supported as a particular case of WTG. WTG follows the predecessor-take-precedence policy with some refinements and a security level policy. Also, it proposes a new conflict resolution method, *strict-predecessor-take-precedence*. It means that the owner of the resource establishes a hierarchy of subjects and any of the further delegations made for these subjects has to preserve this hierarchy. For instance, if A gets from S the higher priority in the hierarchy, all his statements take preference over the others ones.

The aim of this paper is to present a graphical representation of both authorization policies and the trust on delegation and authorization credentials. It also elaborates on the idea of acceptance levels for authorizations, defining new indexes that take into account only a given percentage of credentials (but not all of them). We give some guidelines for the definition of authorization policies and provide, as mentioned, a graphical representation.

The rest of the paper has the following structure. In Section 2 we present a revision of WTG, with some interesting changes with respect to the original work. Section 3 offers new definitions that make WTG more suitable for certain environments, providing alternatives to the original definitions. Section 4 focuses on the concept of authorization policies and we provide an original graphical representation of them. Section 5 ends with some conclusions.

2 Weighted Trust Graph

In this section we examine WTG, overviewing its main ideas, what will help to understand better the concepts that will be explained in the following sections.

In WTG, credentials are represented using arcs in a graph, so both terms are used likewise. A credential is a 4-tuple: $(Issuer, Subject, Type, Right)$, where the first component is the issuer of the authorization or delegation statement; the second one is whom the statement refers to; the third is the type of the statement; and, finally, the fourth is the right together with the resource we are reasoning about.

In fact, $Right$ can be represented as a 2-tuple consisting of the resource and the kind of access, thus $Right = (Resource, Access)$. It must be noted that $Type$ can be expressed as a 3-tuple composed of the following parameters:

- *Weight*, which represents the level of trust in this authorization. It is a number in the interval $[0, 1]$. A credential with weight zero is equivalent to a null credential.
- *Delegation*, which represents whether it is a delegation statement or not.
- *Sign*, which represents the sign of the statement (either negative or positive). Negative credentials may override positive ones and the other way around, depending on the weight.

According to that presentation, WTG defines four types of credentials: positive delegation, positive authorization, negative delegation and negative authorization, that are graphically represented in figure 1.

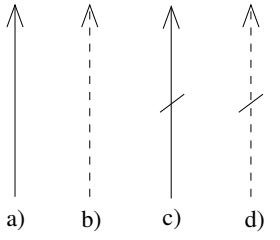


Fig. 1. Representation of credentials

Suppose we are reasoning about an attribute a and we have two principals involved: Alice, the grantor or issuer of the statement and also the owner of the administrative right over attribute a, and Bob, the subject of the statement.

The simplest type of credential corresponds to the concept of Authorization. In this situation Alice owns attribute a and she can issue statements regarding attribute a and attribute $\neg a$. It is important to note that, even if a and $\neg a$ are different attributes, they are very related so that the authority about a should imply the authority about $\neg a$. For instance, let's think about the case of the attributes $Female$ and $Male$. They are complementary attributes, which means that $Male := \neg Female$. In this case, positive authorization regarding $Male$ is equivalent to a negative authorization regarding $Female = \neg Male$.

As a result, although we can easily include both attributes a and $\neg a$ in an authorization chart using positive and negative authorizations, we can not do it in a delegation chart because we would require six different arcs. This leads us to the situation of having a more simple graphical representation with only the four types of statements depicted in Figure 1, but with a meaning different to the original WTG:

a) *Positive delegation.* A positive delegation about both a and $\neg a$.
b) *Positive authorization.* A positive authorization about a or a negative Authorization about $\neg a$.
c) *Negative delegation.* A negative delegation about both a and $\neg a$.
d) *Negative authorization.* A negative authorization about a or a positive Authorization about $\neg a$.

When making decisions regarding authorization to perform certain operations over a resource, it is necessary to consider all the chains or paths of credentials from the owner of the resource to the specific subject. WTG defines paths of credentials as sequences of consecutive credentials, distinguishing between *delegation paths* (those in which there are only delegation credentials), and *authorization paths* (delegation paths followed by a single authorization credential). Only credentials regarding the same Right can be chained, otherwise the resulting path makes no sense.

WTG defines *metrics* over paths. Those metrics help us to measure the relative authorization power of different paths. It only uses monotone metrics (for motivation, see [1]). Some examples of metrics are:

- $|C|_{\cdot} = |m_1||m_2| \cdots |m_n|$
- $|C|_{min} = min(|m_1|, |m_2|, \ldots, |m_n|)$
- $|C|_{+} = |m_1| + |m_2| + \ldots + |m_n|$
- $|C|_{max} = max(|m_1|, |m_2|, \ldots, |m_n|)$

where each m_i represents an arc/credential in the path C and $|m_i|$ refers to the weight of the arc/credential.

Depending on the domain of the possible values for Weight the metrics previously defined are increasing or decreasing functions. In particular, the metric $(|C|_{+}, \mathbb{N})$ is that one used by Ruan et al. in [4]. Note that Ruan define the weight 0 as the higher certainty level, so a higher value for $|C|_{+}$ represents a "weaker" path.

However, WTG takes the opposite approach by using $(|C|_{\cdot}, [0, 1])$ as the metric. In this case, a lower weight represents a "weaker" path ("weak" means that if the path C' is weaker than C, then it should be overridden by C).

The definition of metrics is the key for conflict resolution and allows to measure the priority of each authorization, or at least to compare them. Although there are a variety of orders that can be defined using metrics, others can not. One example is the lexicographic or dictionary order, denoted by $<_l$. In this case, the lexicographic order refers to the weight of the arcs in a path. When comparing two paths using the lexicographic order, we start by the closer arc from the owner of the resource and compare them until we find two arcs with different weights. At this stage, the path with the lower weight in this arc is overridden by the other path.

Given a metric, $|\cdot|$, over paths WTG defines $\mathcal{H}_{AB}$ and $\mathcal{L}_{AB}$ as the maximum and minimum weight, respectively, over all the authorization paths from A to B regarding the same Right. In other words, $\mathcal{H}_{AB}$ is the weight of the "better" path and $\mathcal{L}_{AB}$ is the weight of the "worst" path. That will help us to define the authorization policies. Note that all the indexes are meaningless without the definition of an associated policy. Authorization policies will be defined in section 4.

2.1 Re-visiting the Mean Index Concept

WTG original work defined the average weight (Mean index), $\mathcal{M}_{AB}$, for softening the differences between $\mathcal{H}_{AB}$ and $\mathcal{L}_{AB}$. Although this index is very useful, it was relegated to a second level in that work.

Now, we further elaborate on this concept, providing an algorithmic definition. $\mathcal{M}_{AB}$ will be considered as a graph exploration using a *branch and bound* alike algorithm in which we incrementally calculate $\mathcal{M}_{AX}$ for each node X that is in a path from A to B.

We initialize $\mathcal{M}_{AX} = 0$ for all $X \neq A$ and $\mathcal{M}_{AA} = 1$, and associate those values to the corresponding nodes. In order to calculate $\mathcal{M}_{AB}$, it is necessary to inspect in the first step the principals connected from A with a single arc (*branch* phase), and add the weight of the corresponding arc to the weight of the node.

Then, the negatives nodes are marked as "non useful" because they can not further delegate, thus can not be part of any delegation path (*bound* phase). The process is repeated until B is reached. The result is that all non-useful nodes are marked. When reasoning about two principals A and B, the non useful nodes are omitted and an effective graph containing only the useful nodes is obtained. The resulting graph is easier to inspect, both visually and arithmetically. Each time a new arc (credential) is added to the system, we have to update the values calculated previously. As a consequence, valid nodes may be turned into negative ones (see Figure 2 for example). A positive delegation arc may imply a positive authorization arc, i.e. nodes which receive a delegation arc may issue an authorization arc pointing to themselves. Then, a negative authorization arc is in conflict with a positive delegation.

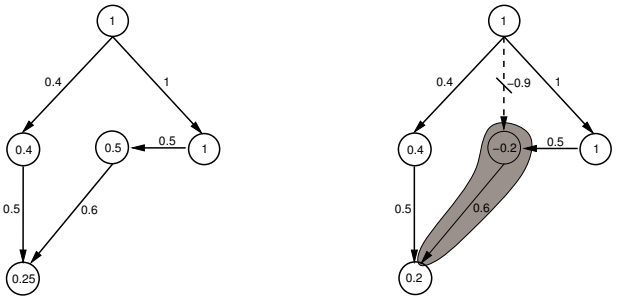


Fig. 2. Update of Trust Graph

3 Improving Authorization Representation Through Graphics

In this section we present important improvements to WTG that are of great value to provide a general solution. That is, these improvements provide alternatives to organizations where traditional authorization systems do not fit well. We also propose here a graphical representation of the authorization strength, defined as the level of trust that the issuer of the statement gives to the subject. This graphical representation is specially useful for those organizations that combine human and computer decisions.

One of the problems that we faced in the initial WTG work is that the Mean Index was affected negatively by low paths. Hence, adding a new path with a very low index leaded to a very low Mean Index. The main reason is that the weights of the credentials were not always normalized.

There is a basic relation among the idexes defined in WTG,

$$-1 \leq \mathcal{L}_{AB} \leq \mathcal{M}_{AB} \leq \mathcal{H}_{AB} \leq 1$$

$\mathcal{M}_{AB}$ is a index that offers us an average information about the authorization; however, the computation involved is very hard. On the other hand, $\mathcal{H}_{AB}$ and $\mathcal{L}_{AB}$ are less helpful but the computation involved is very simple. Thus, we have to combine them to reach a balanced solution.

We now present some methods to represent graphically those indexes. Given two principals A and B, we can represent $\mathcal{L}_{AB}$ and $\mathcal{H}_{AB}$ in the box $[-1,1] \times [-1,1]$ using the point $(\mathcal{H}_{AB}, \mathcal{L}_{AB})$. Moreover, as $\mathcal{L}_{AB} \leq \mathcal{H}_{AB}$, the point should be in the triangle of vertexes $(-1,-1),(1,-1),(1,1)$, as shown in Figure 3.a.

Then, the Mean can be represented as the point $(\mathcal{M}_{AB}, \mathcal{M}_{AB})$ in the region $\mathcal{H}_{AB} \geq X$,$Y \geq \mathcal{L}_{AB}$. Once we know how to represent the Mean in the axis, we may think of making authorization decisions based only in this index. However, the Mean is a more complex index regarding calculation, and there are cases in which this means that is not a good representative of the set of path weights. We could do it only when the Mean is a "good" representative of the whole set of path weights.

From the theory of statistics, we know that it is necessary to take a look at the standard deviation, what informs us how tightly all the weights are clustered around the Mean. Then, a low deviation informs us that the Mean is a "good" mean, while a high deviation represents that there are too many contradictory statements, so in this case the mean can not be considered as a measure of the trust strength between A and B and we should define a different mechanism.

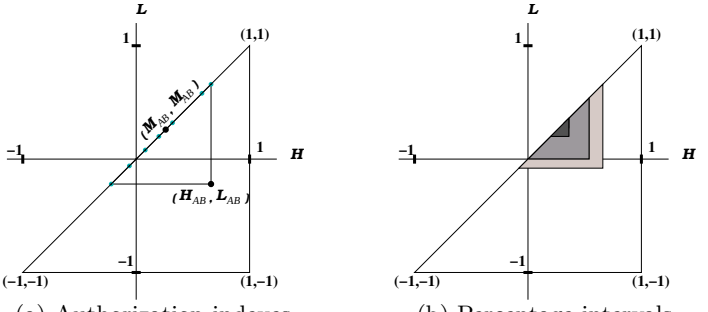
In order to avoid computing the deviation, we could represent every path weight with the point (w,w), where w is the weight of the path. In figure 3.a we represent the weights as grey filled points and the mean as a bigger dark point. The representation of these points allow us to know the density of the path weights around the mean and the indexes $\mathcal{L}_{AB}$ and $\mathcal{H}_{AB}$ together in the same axes diagram.

Although figure 3.a gives enough information regarding how good is the Mean Index, we can simplify this chart in the following way. We define a function that provides, for a given percentage x, the smallest interval centered in the Mean Index which contains at least the x percentage of weights.

Definition 1. *Let A, B be two entities and*

$$r_x := min\{y \in \mathbb{R} : [\mathcal{M}_{AB} - y, \mathcal{M}_{AB} + y] \text{ contains the } x\% \text{ of weights}\}$$

Then, we define the ***x-percentage interval*** *as $[\mathcal{L}^x_{AB}, \mathcal{H}^x_{AB}]$, where $\mathcal{L}^x_{AB} := max\{\mathcal{L}_{AB}, \mathcal{M}_{AB} - r_x\}$ and $\mathcal{H}^x_{AB} := min\{\mathcal{H}_{AB}, \mathcal{M}_{AB} + r_x\}$*



(a) Authorization indexes (b) Percentage intervals

Fig. 3. Graphical representation

As a particular case, $\mathcal{H}_{AB}^{100\%} = \mathcal{H}_{AB}$ and $\mathcal{L}_{AB}^{100\%} = \mathcal{L}_{AB}$. We can now represent those intervals for the cases of x equal to 50%, 75% and 100%, respectively, using the triangle of vertexes $(\mathcal{L}_{AB}^{x}, \mathcal{L}_{AB}^{x})$, $(\mathcal{H}_{AB}^{x}, \mathcal{L}_{AB}^{x})$, $(\mathcal{H}_{AB}^{x}, \mathcal{H}_{AB}^{x})$. We fill them with different grey color scales, considering that the darker is the color, the lower is the percentage. We show a sample representation in figure 3.b. If the deviation is low, then the darker triangles will become small and, on the contrary, if the deviation is high then the darker triangles will become big.

4 Associated Policies

The owner of the resource or attribute should be able to define different authorization requirements depending on the situation and on the resource or attribute. Note that there could be some critical resources that require a more restrictive authorization policy, but also other situations in which a non-critical object become critical and the associated security policy has to be changed. By separating the definition of the credential from the definition of the policies we obtain a more flexible authorization system. In this sense, we mean by *policies* the way of determining, according to the weights of each credential, if one entity is authorized by another one to perform an operation.

What we propose in the previous paragraph would allow us to change the policy according to the credentials defined, and the other way around. In this section we propose different authorization policies that can be used separately, or grouped.

One possible authorization policy would be to define an acceptance interval for the Mean Index. However, we start defining simpler policies which only depends on the simplest indexes $\mathcal{H}_{AB}$ and $\mathcal{L}_{AB}$ because they are more efficient in computation time and complexity.

The simplest policy we can think about is to grant the operation if there is any positive path between the owner of the resource and the subject. The major drawback of this policy is that positive and negative paths can be in conflict because we also use negative paths. The case *there is a positive path from A to B* translates in our formalism to $\mathcal{H}_{AB} > 0$. This is a must for authorization,

but we need more restrictive conditions. A high restrictive approach would be to impose $\mathcal{L}_{AB} > 0$, that translates to *there are no negative paths from A to B*.

If we assign to each pair of entities A, B the bidimensional vector $(\mathcal{H}_{AB}, \mathcal{L}_{AB})$ as done in the previous section, we can represent a policy in an axes diagram as a subset of the triangle of vertexes $(-1,-1)$, $(1,-1)$ and $(1,1)$ that include all the acceptable tuples $(\mathcal{H}_{AB}, \mathcal{L}_{AB})$ for that particular operation to be granted.

Once explained the concept of authorization policy, we examine how a policy set P looks like. Suppose that we have a point inside P, i.e. let A, B be two entities such that $(\mathcal{H}_{AB}, \mathcal{L}_{AB}) \in P$. If A', B' are two entities with $\mathcal{H}_{A'B'} \geq \mathcal{H}_{AB}$ and $\mathcal{L}_{'A'B} \geq \mathcal{L}_{AB}$, this means that the lower path from A' to B' is greater than the lower path from A to B. It occurs similarly with the greater path. So we conclude that the paths from A' to B' are "better" than the ones from A to B. As a consequence, the point $(\mathcal{H}_{A'B'}, \mathcal{L}_{A'B'})$ should be in P too.

Definition 2. *We define a bound policy as a subset $\mathcal{P}$ of the triangle of vertexes $(-1,-1)$, $(1,-1)$ and $(1,1)$ with the following property: $(x,y) \in P$ implies that $(x',y') \in P$ for all $x' \geq x, y' \geq y$.*

We say that B is granted authorization from A if $(\mathcal{H}_{AB}, \mathcal{L}_{AB}) \in \mathcal{P}$.

The next authorization policy we include in our solution is based on Definition 2. In this case, we relax the condition of $(\mathcal{H}_{AB}, \mathcal{L}_{AB}) \in \mathcal{P}$ by allowing a few number of path weights to be out of P.

Definition 3. *Given a policy set P, we say B is granted access from A according to the x-Percent Policy for the set P, if the x percent of path weights are in P.*

In particular, if $(\mathcal{H}^x_{AB}, \mathcal{L}^x_{AB}) \in P$ then B is granted by A at x percent.

Based on Definition 2, we define two example policies in which we always force $\mathcal{H}_{AB} > 0$:

- **Absolute bound policy:** we choose a lower bound K for the lower path between two nodes. Only entities with $\mathcal{L}_{AB} > K$ will be authorized by this policy. We may define the policy set formally as

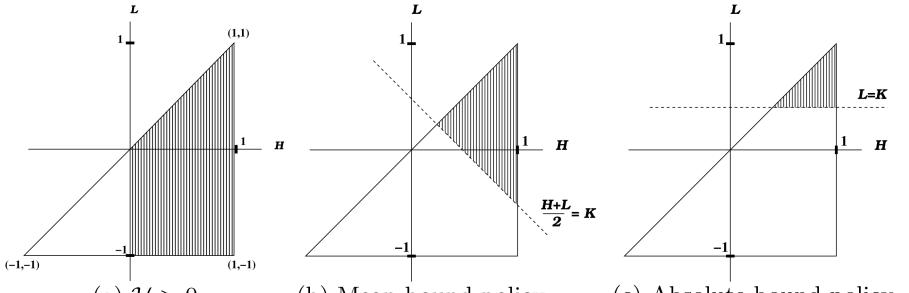
$$P := \{(\mathcal{H}, \mathcal{L}) : \mathcal{L} > K\}$$

- **Mean bound policy:** we choose a lower bound K for the mean of $\mathcal{H}_{AB}$ and $\mathcal{L}_{AB}$. In this policy, a positive path overrides a lower negative path. A particular case is when K is equal to zero. Formally,

$$P := \{(\mathcal{H}, \mathcal{L}) : \mathcal{H} + \mathcal{L} > 2K\}$$

In figure 4, we represent some policies in an axis diagram.

In a Mean bound policy, when $K = 0$, it could happend that $\mathcal{H}_{AB} = -\mathcal{L}_{AB}$. Then, we use the lexicographic order to decide if we grant authorization or not by using the following procedure: we authorize entities if any of the paths with the highest weight is greater (using the dictionary order) than all the paths with the lowest weight. In other words, if there exits a path C with $\mathcal{H}_{AB} = |C|$ and $C >_L C'$ for all C' with $|C'| = \mathcal{L}_{AB}$



(a) $\mathcal{H} > 0$ (b) Mean bound policy (c) Absolute bound policy

Fig. 4. Different types of policies

We use the lexicographic order to solve the case in which $\mathcal{H}_{AB} + \mathcal{L}_{AB} = 0$ but we can use the lexicographic order alone to decide about authorization. This is the reason why we define a **lexicographic policy** or hierarchical policy. We order all the paths from A to B according to the dictionary order and, if the maximal elements are all positives, then B is authorized. In case there are maximal negative paths, authorization is denied.

We define a third kind of authorization policy that we name **security level policy**. We define a real number K, as in the bound policy, but we use it to discard credentials with weight lower than K. Thus, we refine the delegation graph after computing the indexes, discarding all the credentials with a "low" weight. After this, we should apply some of the previous policies to decide about the authorization. We say that a credential path is *k-valid* if all the arcs in the path have weight greater or equal than k.

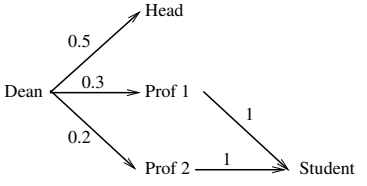


Fig. 5. Security level policy

Suppose that we have three different security levels in the system represented in figure 5. For each security level we choose a different K. In this case, let $K_1 = 0.2$ be the lower security level, $K_2 = 0.3$ the second one and $K_3 = 0.5$ the highest level. In order to show how it works we use students authorizations:

- **Level 1** (0.2)**.** In this security level, Student gets access to the resource, because Professor 2 issued a 0.2-valid statement.
- **Level 2** (0.3)**.** In this security level, Student gets access to the resource, but Professor 2 authorization is not enough because the path goes across himself if not 0.3-valid. Student needs to use the statement issued by Professor 1.

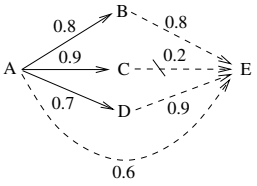


Fig. 6. Example of delegation/authorization graph

– **Level 3** (0.5)**.** In this security level, Student gets no access to the resource, because there is no any path higher than 0.5 that allows him to access the resource. The only one who can issue such a statement is the Dean.

4.1 Example of Usability

Consider the example of Figure 6 in which we have five entities $\{A, B, C, D, E\}$ and seven credentials. We focus on the relations between A and E.

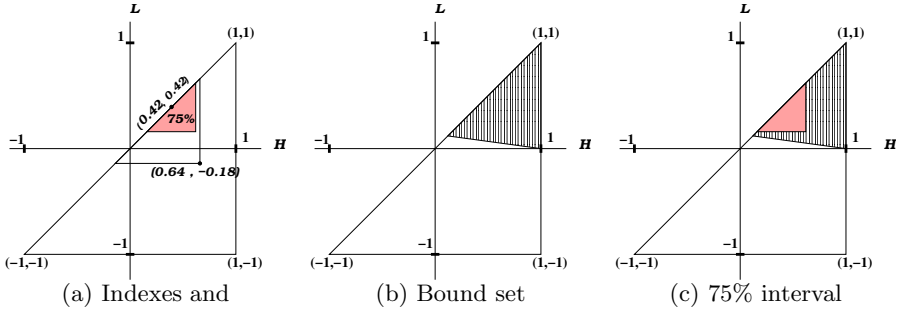
In this case, $\mathcal{H}_{AE} = 0.64$, $\mathcal{L}_{AE} = -0.18$ and $\mathcal{M}_{AE} = 0.4225 \sim 0.42$. Then we calculate the 75% interval for the Mean. We first calculate $r_{75\%}$

$$r_{75\%} := min\{y \in \mathbb{R} : [0.42 - y, 0.42 + y] \text{ contains the } 75\% \text{ of weights}\}$$

$r_{75\%} = 0.64 - 0.42 = 0.22$ so, $\mathcal{L}_{AE}^{75\%} := max\{-0.18, 0.20\} = 0.20$ and $\mathcal{H}_{AE}^{75\%} := min\{0.64, 0.64\} = 0.64$.

The graphical representation of those indexes is in Figure 7. The grey triangle represent the 75% interval for the Mean. In this diagram we see that the area of the grey triangle is more or less half the area of the big triangle, what indicates that the weight of the paths are clustered around the mean. Hence, the Mean is a good representative for all the weights.

If we remove the negative path we get a very small triangle. Then, the smaller the triangle is, the better representative the Mean Index is. Looking at Figure 7.a we conclude that the Mean Index really represents the overall weights. A computer is also able to reach the same conclusion by inspecting $r_{75\%}$ (the smaller it is, the better representative the mean is).



(a) Indexes and 75% interval (b) Bound set (c) 75% interval inside bound set

Fig. 7. Sample bound policy

The next step is to define different authorization policies for the previous example.

With this information we propose different authorization policies. In the example, if we propose a triangle P with their vertexes in the first quadrant (which means that we do not allow negative credentials for granting authorization) of the axes, the associated bound policy is that we will deny the authorization to E because $(0.64, -0.18)$ is not in the first quadrant. If we relax these conditions and opt for a Percent Policy of 75%, we may define several sets P in the first quadrant, what will lead us to grant the authorization (see Figure 7.b., 7.c.).

Therefore, we choose if the resource is so critical as to avoid negative credential or if we allow them but in case the Mean is good enough. In case we decide to use the lexicographic policy or hierarchical policy we will use the negative path to decide about authorization because it has the higher weight over all the first arcs in all paths. Choosing the lexicographic policy will lead to a denial of authorization from A to E.

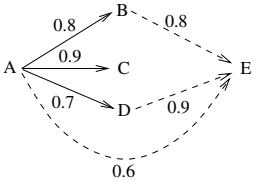


Fig. 8. Delegation graph after applying the security level policy

Let's suppose that we apply a security level policy to discard non-relevant credentials (those which has a "small" weight). If we choose a bound lower than 0.2 then all credentials remain in the system. But if we choose 0.5 as a security level bound, we will avoid the negative path (see Figure 8). After that we could apply any of the previous policies to decide about authorization.

The application of the security level policy changes dramatically the situation. If we calculate the indexes for the new graph represented in Figure 8 we get $\mathcal{H}_{AE} = 0.64$, $\mathcal{L}_{AE} = 0.6$ and $\mathcal{M}_{AE} = 0.623\ldots \sim 0.62$. In this case, the Mean Index is very accurate (indeed $r_x \leq 0.02$ for all percentage intervals). We can compare the two situations according to r_x: in the original example, $r_{100\%} = 0.6$ and $r_{75\%} = 0.22$; now $r_{100\%} = 0.02$ and $r_{75\%} = 0.016\ldots$. Note that smaller intervals represent more clustered values.

5 Conclusions

We have presented in this work a major extension for WTG which comprises three main issues: definition of new indexes, graphical representation of indexes, and graphical representation of authorization policies.

Because our goal was to integrate this work into a user-oriented application, we provide a graphical representation of the indexes that is helpful for security administrators in defining authorization policies.

The graphical representation of indexes has opened the door to the definition of a graphical representation of policies. Having a graphical representation of authorization policies allows human decisions to be better included in the system. With logic-based frameworks, human interaction in the decision phase is impossible. However, in our framework, and because the representation of both policies and authorization (delegation) credentials is graphical, humans can interact based on this graphical information.

References

1. Isaac Agudo, Javier Lopez and Jose A. Montenegro. A representation model of trust relationships with delegation extension. In *3rd International Conference on Trust Management, iTrust 2005*, volume 3477 of *Lecture Notes in Computer Science*, pages 116 – 130. Springer, 2005.
2. Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.
3. Chun Ruan and Vijay Varadharajan. Resolving conflicts in authorization delegations. In *ACISP'02*, volume 2384 of *Lecture Notes in Computer Science*, pages 271 – 285. Springer, 2002.
4. Chun Ruan and Vijay Varadharajan. A weighted graph approach to authorization delegation and conflict resolution. In *ACISP'04*, volume 3108 of *Lecture Notes in Computer Science*. Springer, 2004.
5. Chun Ruan, Vijay Varadharajan, and Yan Zhang. Logic-based reasoning on delegatable authorizations. In *ISMIS '02: Proceedings of the 13th International Symposium on Foundations of Intelligent Systems*, volume 2366 of *Lecture Notes in Computer Science*, pages 185–193. Springer, 2002.
6. Chun Ruan, Vijay Varadharajan, and Yan Zhang. A logic model for temporal authorization delegation with negation. In *6th International Information Security Conference, ISC 2003*, volume 2851 of *Lecture Notes in Computer Science*, pages 310 – 324. Springer, 2003.

Secure Cross-Realm C2C-PAKE Protocol

Yin Yin[1] and Li Bao[1,⋆]

State Key Laboratory of Information Security
Graduate School of Chinese Academy of Sciences
Beijing 100049, China
yin@is.ac.cn, lb@is.ac.cn

Abstract. Client-to-client password authenticated key exchange (C2C-PAKE) protocol deals with the authenticated key exchange process between two clients, who only share their passwords with their own servers. Jin Wook Byun *et al.* first divided this scenario into two kinds called single-server C2C-PAKE protocol and cross-realm C2C-PAKE protocol respectively. Recently, Abdalla *et al.* proposed a generic construction for single-server C2C-PAKE protocol and presented a concrete example with security proof. But, no similar results about cross-realm C2C-PAKE protocol exist. In fact, all existing cross-realm C2C-PAKE protocols are found insecure. To counter flaws and provide a secure cross-realm C2C-PAKE protocol, in this paper, we introduce a formal model and corresponding security definitions. Then, a new cross-realm C2C-PAKE protocol is presented with security proof.

1 Introduction

Key exchange protocol is a sophisticated topic in cryptographical research community. From the seminal paper of Diffie and Hellman [12], many such protocols have been proposed. The plain form of Diffie Hellman protocol, however, lacks necessary authentication. To address this problem, some prior information among participants is required to authenticate their identities, either a common long term key or a shared short password.

Compared with long term key, password is more preferred because its short length facilitates humans to remember it in minds. So a roaming user can authenticate himself easily without any assistant devices and put the security only on the password. But on the other hand password's short length, which means low entropy (usually 6-8 ASCII characters), is also the major drawback of the password-based protocol. A secure password-based protocol must resist the so called *off-line dictionary attack*, which enables the adversary to search the password in the dictionary off-line.

Now, most password-based authenticated key exchange protocols (PAKE) only consider how to establish a session key between two participants who have shared a common password in advance, such as PAK protocol suits [8, 14, 15]

⋆ This research is supported by the National Natural Science Foundation of China under Grant No.90304013.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 395–406, 2006.
© Springer-Verlag Berlin Heidelberg 2006

and AuthA protocol suits [4, 9, 10, 1]. Following these protocols, if there are n participants and we want to provide secure communication between every pair of them, everyone has to remember $n-1$ different passwords, which is inefficient and impractical. In order to reduce the number of passwords and improve efficiency, some client-to-client password authenticated key exchange (C2C-PAKE) protocols have been proposed, such as [11, 13, 17, 16, 2, 3].

To our knowledge, Jin Wook Byun et al. [11] firstly considered and divided client-to-client password authenticated key exchange protocol into two types: single-server C2C-PAKE protocol and cross-realm C2C-PAKE protocol.

Single-Server C2C-PAKE Protocol: In this setting, a single universal trusted third party (TTP) is involved. All clients share their passwords with the TTP rather than share a password between every pair of clients. So in the single-server C2C-PAKE protocol, every client only need remember his own password which enables him to establish a session key with another client through the TTP.

Recentlly, Abdalla et al. [2] formally addressed this setting and provided a generic method to construct provably secure single-server C2C-PAKE protocol. To reduce the generic construction's complexity and provide a concrete protocol, the first provable secure single-server C2C-PAKE protocol is proposed in [3].

Cross-Realm C2C-PAKE Protocol: Compared with normal PAKE protocol, single-server C2C-PAKE protocol provides an efficient solution for peer communication in a large group. But due to the single universal TTP assumption, this solution's application is limited. When all clients belong to the same corporation or the same university, the single universal TTP may be available. However, when clients come from different organizations, it is a natural desire not to put the whole system's security on a single TTP. In fact, every organization would have their own trusted servers to manage members' passwords and provide service for them. If a client shares his password with a server, we say that the client is in the realm of the server. So every server stores the passwords of all clients belonging to his realm.

In the two-party password-based protocol, it only needs to ensure that any outside adversary cannot gain information about the session key or run a successful *off-line dictionary attack* on the password. But in the C2C-PAKE protocol, a legal client may also try to learn other client's password after the execution of the protocol. We call this attack as *client's inside attack*. Furthermore, for cross-realm C2C-PAKE protocol, a legal server may also try to learn the password of the client not belonging to his realm. We call this attack as *server's inside attack*. So a secure cross-realm C2C-PAKE protocol means that it does not reveal any information about the session key and the password under *off-line dictionary attack*, *client's inside attack* and *server's inside attack*.

1.1 Existing Cross-Realm C2C-PAKE Protocols

The first cross-realm C2C-PAKE protocol is proposed by Jin Wook Byun et al. in [11], denoted as C2C-PAKE-BJLP. Later, in [17], Shuhong Wang *et al.* found that a passive server could learn enough information to run an *off-line dictionary attack* on a client who does not belong to his realm. As a result, C2C-PAKE-BJLP is vulnerable to *server's inside attack*. To eliminate this flaw, Shuhong Wang *et al.* proposed a fix, denoted as C2C-PAKE-WWX.

But quickly in the same year, Jeeyeon Kim *et al.* [13] found a *client's inside attack* which could be applied to both C2C-PAKE-BJLP and C2C-PAKE-WWX. With this attack, a malicious client could learn other client's password in the protocol. Again, to avoid attacks found by them, Jeeyeon Kim *et al.* proposed a new cross-realm C2C-PAKE protocol, called C2C-PAKE-KKKW.

Unfortunately, this new protocol C2C-PAKE-KKKW is also insecure found later by Raphael Chung-Wei Phan *et al.* in [16]. Chung-Wei Phan *et al.* pointed out that C2C-PAKE-KKKW is vulnerable to *off-line dictionary attack*, *client's inside attack* and *server's inside attack*. But they did not provide any fix.

Hitherto, there has been a long sequence of works proposing protocols for cross-realm PAKE without any proof of security which are subsequently broken. All existing protocols are summarized in the Table 1.

Table 1. The comparisons of existing cross-realm C2C-PAKE protocols. ✓ denotes secure, × denotes insecure.

	off-line dictionary attack	server's inside attack	client's inside attack
C2C-PAKE-BJLP [11]	✓	×	×
C2C-PAKE-WWX [17]	✓	✓	×
C2C-PAKE-KKKW [13]	×	×	×

1.2 Organization

In Section 2, we describe our construction of the new cross-realm C2C-PAKE protocol. To prove its security, we first describe our model and necessary basic assumptions in Section 3. In Section 4, we discuss the security of our new protocol and provide details of the proof. Finally, Section 5 concludes our paper.

2 Our Cross-Realm C2C-PAKE Protocol

In this section, we present our cross-realm C2C-PAKE protocol. Our protocol could be viewed as an extension of the protocol proposed in [3], which is designed for single-server C2C-PAKE. We extend the single server in [3] to two separate servers and carefully define their inner communication. But for the outside adversary or the client, all messages obtained in our new protocol and those obtained in the protocol of [3] are exactly the same.

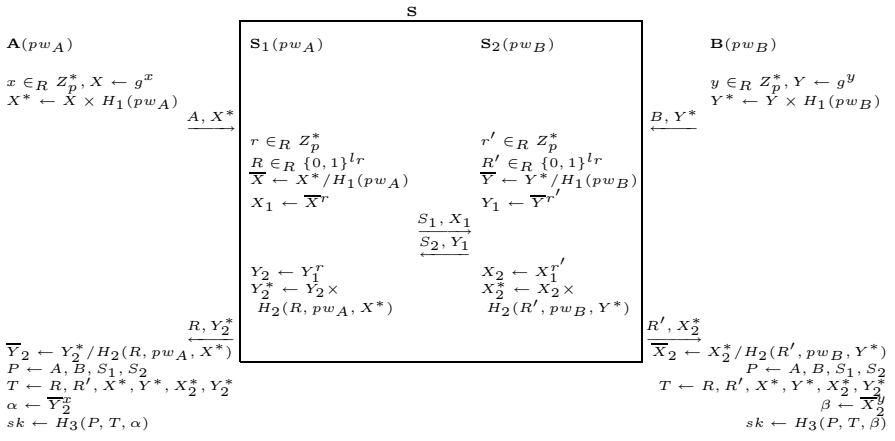


Fig. 1. New C2C-PAKE protocol

Four participants are involved in our protocol, denoted as A, S_1, B, S_2 respectively, where A is a client in the realm of server S_1, B is a client in the realm of server S_2, and there is an authenticated private communication channel between S_1 and S_2. In practice, the authenticated private communication channel can be implemented by the key shared between servers in advance or their public keys. The communication model and work flows are depicted in Figure 1, where H_1, H_2, H_3 are three hash functions modeled as random oracles, l_r is a security parameter.

Our construction consists three phases:

Phase 1. In this phase, client A and client B both choose their random input x, y from Z_p^*. They compute $X = g^x, Y = g^y$ and blind them as $X^* = X \times H_1(pw_A), Y^* = Y \times H_1(pw_B)$. At the end, A sends X^* to S_1 and B sends Y^* to S_2.

Phase 2. Upon receiving X^* and Y^* from clients, *Server*s reblind them and exchange with each other. S_1 computes $\overline{X} = X^*/H_1(pw_A)$ and reblinds it as $X_1 = \overline{X}^r$, where r is S_1's first random input from Z_p^*. S_2 also chooses his random input r' from Z_p^* and computes Y_1 similarly. Then S_1 and S_2 exchange X_1 and Y_1. After that, S_1 computes $Y_2 = Y_1^r$ and $Y_2^* = Y_2 \times H_2(R, pw_A, X^*)$, where R is S_1's second random input. S_2 performs similar operations and gets X_2^*. Finally, S_1 returns R, Y_2^* to client A and S_2 returns R', X_2^* to client B.

Phase 3. In this phase, client A and client B compute their session key sk. Client A first computes $\overline{Y}_2 = Y_2^*/H_2(R, pw_A, X^*)$ and the Diffie-Hellman key $\alpha = \overline{Y}_2^x$. Then the session key is computed from α and the transcript, $sk = H_3(A, B, S_1, S_2, R, R', X^*, Y^*, X_2^*, Y_2^*, \alpha)$. Similar with the client A, client B also computes $\overline{X}_2 = X_2^*/H_2(R', pw_B, Y^*)$ and the Diffie-Hellman key $\beta = \overline{X}_2^y$. Then he also computes the session key $sk = H_3(A, B, S_1, S_2, R, R', X^*, Y^*, X_2^*, Y_2^*, \beta)$. Note that the transcript of the protocol is public. So client A and the client B could collect it by themselves or receive it from servers.

3 Formal Model for Cross-Realm C2C-PAKE Protocol

The first security model about key exchange protocol is proposed by Bellare and Rogaway in [6, 7]. And then in 2000, Bellare *et al.* extended their model to password-based key exchange protocol [5]. Recently, Abdalla, Fouque and Pointcheval introduced a formal model for single-server C2C-PAKE protocol in [2]. Later, this model was used in [3] to prove a concrete single-server C2C-PAKE protocol. In this section, we follow Abdalla *et al.*'s formal model to consider cross-realm C2C-PAKE protocol and formally define the special security requirements of cross-realm C2C-PAKE protocol.

3.1 Communication Model

We denote A, B as two clients belonging to two different realms. Client A shares his password pw_A with server S_1, and client B shares his password pw_B with another server S_2. Additionally, we assume there is an authenticated private channel between server S_1 and server S_2. In practice, this can be implemented by a pre-distributed common key shared between S_1 and S_2 or their public keys. All clients' passwords are chosen from the same small dictionary $\boldsymbol{\mathcal{D}}$ whose distribution is $\boldsymbol{\mathcal{D}_{pw}}$. Each participant may has several instances involved in the execution of protocol. We denote participant U's (maybe a client or a server) i-th instances as U^i. We denote the set of all clients as $\boldsymbol{\mathcal{U}}$, and denote the set of all servers as $\boldsymbol{\mathcal{S}}$.

The cross-realm PAKE protocol is an interactive protocol among four participants' instances: A^i, B^j, S_1^s, S_2^t. After the protocol, A^i and B^j establish a session key sk. During the execution of protocol, an adversary $\mathcal{A}$ could interactive with protocol participants via several oracle queries, which model adversary's possible attacks in the real execution. All possible oracle queries are listed in the following:

- $Execute(A^i, B^j, S_1^s, S_2^t)$: This oracle query is used to simulate eavesdropping attack of the adversary. After querying the oracle, all messages interacted between participants during an execution of the protocol are returned to the adversary. But it should be noted that we assume the communication channel between servers are private, so this oracle query does not reveal communication messages between servers to the adversary unless the adversary is a server.
- $Reveal(U^i)$: This oracle query is used to simulate the misuse of session keys. After querying the oracle, the client instance U^i's session key is returned to the adversary.
- $SendClient(U^i, m)$: This oracle query is used to simulate active attack against the client. After querying the oracle, a message m is sent to the client instance U^i. At the end, client instance U^i's response is forwarded to the adversary.
- $SendServer(S^i, m)$: This oracle query is used to simulate active attack against the server. After querying the oracle, a message m is sent to the server instance S^i. At the end, server instance S^i's response is forwarded to the adversary.

– $Test(U^i)$: This oracle query is not used to simulate adversary's attack, but to define session key's semantic security. After querying the oracle, the session key of U^i or a random number will be returned according to a predefined random bit b. If $b = 1$, the adversary would learn the session key of U^i; otherwise the adversary only learns a random number with the same length. This query can be called only once.

Partnering: As other formal models [5, 2], the relationship of partnering is also defined through the session identifications (sid). We say U^i and U^j are partners if the following conditions are satisfied: (1) Both U^i and U^j accept; (2) Both U^i and U^j own the same sid; (3) U^i is U^j's partner and vice-verse; and (4) No instance other than U^i and U^j accepts with a partner identification equal to U^i or U^j.

Freshness. We say an instance U^i is fresh if it has accepted and no *Reveal* queries have been made to it or its partner.

3.2 Assumptions

We briefly introduce some assumptions required by our cross-realm C2C-PAKE protocol. Let G be a finite cyclic group of prime order p. Let g be a generate element of G. Denote the tuple $\mathbb{G} = (G, g, p)$ as a represented group. Following, DDH assumption is normal and has been heavily used in many works. PCDDH1 and PCDDH2 assumptions are introduced in [3]. We use them to consider insider attack. More information about these two special assumptions and their relationship with other assumptions can be found in [3].

Decisional Diffie-Hellman Assumption(DDH). Let $\mathbb{G} = (G, g, p)$ be a represented group. DDH assumption means that given (g, g^x, g^y), no probabilistic polynomial time algorithm can distinguish g^{xy} from a random element of G with non-negligible probability. We define $Adv_{\mathbb{G}}^{DDH}(\mathcal{A})$ as the probability that the adversary $\mathcal{A}$ could distinguish g^{xy} from a random element of G, and define $Adv_{\mathbb{G}}^{DDH}(t)$ as the maximum value of $Adv_{\mathbb{G}}^{DDH}(\mathcal{A})$ over all $\mathcal{A}$ with time-complexity at most t.

Password-Based Chosen-Basis Decisional Diffie-Hellman Assumption 1(PCDDH1[3]). In the cross-realm C2C-PAKE protocol, a malicious client or a malicious server would try to learn others' password. If the client B is malicious and his goal is to learn client A's password, we want to ensure that he cannot distinguish $Y_2^* = g^{yrr'} \times H_2(R, pw_A, X^*)$ from a random number. Informally, PCDDH1 assumption means that given $(g^x \times H(pw), g^{xs}, g^y, g^{xys})$, any adversary cannot distinguish g^{ys} from a random element of G with non-negligible probability, where x, s are two private random number and the random number y is chosen by the adversary. We define $Adv_{\mathbb{G},N}^{PCDDH1}(\mathcal{A})$ as the probability that the adversary $\mathcal{A}$ could distinguish g^{ys} from a random element of G, where N is the size of the dictionary of pw.

Password-Based Chosen-Basis Decisional Diffie-Hellman Assumption 2(PCDDH2[3]). We use PCDDH1 assumption when the malicious client B does not modify the client A's input X^*. But if X^* is modified, we need a new assumption - PCDDH2 assumption. PCDDH2 assumption means that given $(g^x, (g^x/H(pw))^s, g^y)$, any adversary cannot distinguish g^{ys} from a random element of G with non-negligible probability, where s is a private random number and the other two random numbers x, y are both chosen by the adversary. We define $Adv^{PCDDH2}_{\mathbb{G},N}(\mathcal{A})$ as the probability that the adversary $\mathcal{A}$ could distinguish g^{ys} from a random element of G, where N is the size of the dictionary of pw.

3.3 Security Definition

A secure cross-realm C2C-PAKE protocol should satisfy four security requirements: (1)the session key cannot be distinguished from a random number by an outside malicious adversary; (2)the server does not know the session key between clients; (3)the client does not know other client's password; and (4)clients' passwords are not revealed to other servers except for their own servers. Formally, we define following four security notions:

Semantic Security Against Malicious Outside Adversary: In the $Test$ query, we require the adversary cannot tell whether the response received from the $Test$ oracle is the session key or a random number. In other words, the adversary cannot guess the random bit b used in the $Test$ query with non-negligible probability larger than $1/2$.

We say the adversary succeeds if he correctly guesses the value of b. Let $Succ^{ake}$ denote the event that the malicious outside adversary succeeds. Let $\mathcal{D}$ be user's password dictionary. For any adversary $\mathcal{A}$, we define his advantage $Adv^{ake}_{\mathcal{D}}(\mathcal{A})$ as

$$Adv^{ake}_{\mathcal{D}}(\mathcal{A}) = 2 \cdot \Pr[Succ^{ake}] - 1$$
$$Adv^{ake}_{\mathcal{D}}(t, R) = \max_{\mathcal{A}}\{Adv^{ake}_{\mathcal{D}}(\mathcal{A})\}$$

where the maximum is over all adversaries with time-complexity at most t and using at most R times oracle queries.

We say P is semantically secure against malicious outside adversary if the advantage $Adv^{ake}_{\mathcal{D}}$ is only negligibly larger than $\mathcal{O}(q_s) \cdot \mathcal{D}_{pw}$, where q_s is the number of all send queries, $\mathcal{D}_{pw}$ is the distribution of the password dictionary.

Key Privacy Against Passive Server: We require that no information about the session key is revealed to the server. Note that the server knows all passwords of his members. So a malicious server is always able to impersonate one of its member and exchange a session key with another client by active attack. As a result, we cannot require a malicious server cannot learn the session key.

The passive server $\mathcal{S}$ could query two oracles: $Execute$ and $Test$. We say $\mathcal{S}$ succeeds if he correctly guesses the value of the random bit b used in the $Test$ query. Let $Succ^{kp}$ denote the event that the passive server succeeds. Let $\mathcal{D}$ be

user's password dictionary. For any passive server $\mathcal{S}$, we define his advantage $Adv_{\mathcal{D}}^{kp}(\mathcal{S})$ as

$$Adv_{\mathcal{D}}^{kp}(\mathcal{S}) = 2 \cdot \Pr[Succ^{kp}] - 1$$
$$Adv_{\mathcal{D}}^{kp}(t, R) = \max_{\mathcal{S}}\{Adv_{\mathcal{D}}^{kp}(\mathcal{S})\}$$

where the maximum is over all adversaries with time-complexity at most t and querying oracles at most R times.

We say the protocol P is key private against passive server if the advantage $Adv_{\mathcal{D}}^{kp}$ is negligible.

Password Protection Against Malicious Client: $Test$ oracle query is used to define the session key's security, which is not considered in current security notion. So the malicious client does not have access to $Test$ query. We say $\mathcal{C}$ succeeds if he successfully learns another client's password. Let $Succ^{pw-mc}$ be the event that the malicious client succeeds. Let $\mathcal{D}$ be user's password dictionary. For any malicious client $\mathcal{C}$, we define his advantage $Adv_{\mathcal{D}}^{pw-mc}(\mathcal{C})$ as

$$Adv_{\mathcal{D}}^{pw-mc}(\mathcal{C}) = Pr[Succ^{pw-mc}]$$
$$Adv_{\mathcal{D}}^{pw-mc}(t, R) = \max_{\mathcal{C}}\{Adv_{\mathcal{D}}^{pw-mc}(\mathcal{C})\}$$

where the maximum is over all adversaries with time-complexity at most t and querying oracles at most R times.

We say P satisfies password protection against malicious client if the advantage Adv^{pw-mc} is only negligibly larger than $\mathcal{O}(q_s) \cdot \mathcal{D}_{pw}$, where q_s is the number of all send queries, $\mathcal{D}_{pw}$ is the distribution of password dictionary.

Password Protection Against Malicious Server: Like the notion of password protection against malicious client, for any malicious server $\mathcal{S}$, we define his advantage $Adv_{\mathcal{D}}^{pw-ms}(\mathcal{S})$ as

$$Adv_{\mathcal{D}}^{pw-ms}(\mathcal{S}) = Pr[Succ^{pw-ms}]$$
$$Adv_{\mathcal{D}}^{pw-ms}(t, R) = \max_{\mathcal{S}}\{Adv_{\mathcal{D}}^{pw-ms}(\mathcal{S})\}$$

where the maximum is over all adversaries with time-complexity at most t and querying oracles at most R times.

We say P satisfies password protection against malicious server if the advantage Adv^{pw-ms} is only negligibly larger than $\mathcal{O}(q_s) \cdot \mathcal{D}_{pw}$, where q_s is the number of all send queries, $\mathcal{D}_{pw}$ is the distribution of password dictionary.

4 Security Proof

In this section, we examine all security requirements proposed in the subsection 3.3 and show they are all met.

Semantic Security Against Malicious Outside Adversary: Because of the high complexity of proof about semantic security against malicious outside adversary, we do not prove our protocol directly but compare it with the protocol designed in [3]. In [3], Michel Abdalla and David Pointcheval designed a single-server C2C-PAKE protocol and provided a proof about its semantic security. It is important to note that the communication between servers is authenticated and private. So for any malicious outside adversary, the communication between servers is totally secure. From this viewpoint, when considering outside adversary, we can treat S_1 and S_2 as a single server. Now it is easy to verify that our protocol and the protocol in [3] are the same. From the proof proposed in [3], our protocol is also semantically secure against malicious outside adversary.

Key Privacy Against Passive Server: A passive server only has access to $Execute$ and $Test$ oracles. Key privacy against passive server is ensured by the following theorem.

Theorem 1. *In our cross-realm C2C-PAKE protocol, a passive server cannot learn the session key between clients as long as the* **DDH** *assumption holds in the group* $\mathbb{G}$*. Formally,*

$$Adv_{\mathcal{D}}^{kp}(t, q_e) \leq 2q_e \cdot Adv_{\mathbb{G}}^{DDH}(t + 8q_e\tau_e)$$

where q_e *represents the number of queries to the oracle* $Execute$*,* τ_e *denotes the exponentiation computational time in* $\mathbb{G}$*.*

Proof. Let $\mathcal{S}_1$ be an adversary against the key privacy with time complexity at most t, who queries $Execute$ at most q_e times. Following, we will construct another adversary $\mathcal{A}$ for DDH assumption using $\mathcal{S}_1$. It means that with the input (g, g^a, g^b, z), $\mathcal{A}$ would guesses whether $z = g^{ab}$.

Firstly, $\mathcal{A}$ initializes all system parameters including all clients' passwords and the random bit b used in the $Test$ query. Additionally, $\mathcal{A}$ chooses an integer t, $1 \leq t \leq q_e$, and supposes the adversary S_1 would query the $Test$ on the t-th $Execute$ query. Then $\mathcal{A}$ runs the adversary $\mathcal{S}_1$ in this environment and provides answers for all oracle queries with $\mathcal{A}$'s input and parameters.

Specially, for $Execute(A^i, B^j, S_1^s, S_2^t)$ query, $\mathcal{A}$ chooses all random inputs x, y, r, r', R, R'. Then $\mathcal{A}$ computes all communication messages and returns them to the adversary $\mathcal{S}_1$.

But for the t-th $Execute$ query, $\mathcal{A}$ computes communication messages based on his input (g, g^a, g^b, z). $\mathcal{A}$ chooses the random input r, r', R, R' and computes $X^* = g^a \times H_1(pw_A)$, $Y^* = g^b \times H_1(pw_B)$, $X_1 = g^{ar}$, $Y_1 = g^{br'}$, $X_2^* = g^{arr'} \times H_2(R', pw_B, Y^*)$ and $Y_2^* = g^{brr'} \times H_2(R, pw_A, X^*)$. Next $\mathcal{A}$ sets the Diffie-Hellman key $\alpha = \beta = z^{rr'}$ and computes the session key as described in the protocol.

For $Test$ query, $\mathcal{A}$ first check whether this $Test$ query is about the t-th $Execute$ query. If it is not in the case, the adversary $\mathcal{A}$ fails; otherwise, $\mathcal{A}$ responses the session key to $\mathcal{S}_1$ when $b = 1$, or responses a random number to $\mathcal{S}_1$ when $b = 0$.

After all interaction, $\mathcal{A}$ sets his answer as the answer of the adversary $\mathcal{S}_1$. Now we analyze $\mathcal{A}$'s advantage. If $z = g^{ab}$, the simulation of $\mathcal{A}$ is perfect. Hence, the probability that $\mathcal{A}$ outputs 1 is exactly $1/2 + 1/2 \cdot Adv_{\mathcal{D}}^{kp}(\mathcal{S}_1)$. If z is a random number, the session key computed in the t-th *Execute* query is also a random number. Hence, the probability that $\mathcal{A}$ outputs 1 is exactly $1/2$. Further note that with probability $1/q_e$, the *Test* query would be called on the t-th *Execute* query. So

$$\begin{aligned} Adv_{\mathbb{G}}^{DDH}(\mathcal{A}) &= \frac{1}{q_e} \cdot (\frac{1}{2} + \frac{1}{2} Adv_{\mathcal{D}}^{kp}(\mathcal{S}_1) - \frac{1}{2}) \\ &= \frac{1}{2q_e} Adv_{\mathcal{D}}^{kp}(\mathcal{S}_1) \end{aligned}$$

The proof of Theorem 1 follows from the fact that $\mathcal{A}$ has time complexity at most $t + 8q_e\tau_e$ because of the additional time for the simulation of *Execute* queries. □

Password Protection Against Malicious Client: In the cross-realm C2C-PAKE protocol, a malicious client may want to learn other client's password. In our protocol, we suppose client B is malicious and his goal is to learn client A's password pw_A. This security notion is ensured by the following theorem.

Theorem 2. *In our cross-realm C2C-PAKE protocol, the malicious client B cannot learn the client A's password as long as the* **PCDDH1** *and* **PCDDH2** *assumptions hold in the group $\mathbb{G}$. Formally,*

$$Adv_{\mathcal{D}}^{pw-mc}(t, q_s) \leq q_s \cdot max\{Adv_{\mathbb{G},N}^{PCDDH1}(t), Adv_{\mathbb{G},N}^{PCDDH2}(t)\} + q_s/N$$

where q_s represents the number of queries to the oracle SendServer and SendClient.

Proof. From the execution of the protocol, the malicious client B chooses the random number y and learns $X^*, g^y, R, g^{yrr'} \times H_2(R, pw_A, X^*), (X^*/H_1(pw_A))^{rr'}$. We would show that $g^{yrr'} \times H_2(R, pw_A, X^*)$ could be replaced by a random number. Furthermore, because rr' is a private random number, X^*, g^y and $(X^*/H_1(pw_A))^{rr'}$ are three independent random numbers from which no information about pw_A is revealed. As a result, the probability that client B correctly guesses pw_A is exactly q_s/N after q_s times send queries, where N is the size of the dictionary.

We replace Y_2^* with a random number. Let **BAD** be the event that client B can distinguish Y_2^* from a random number. So we have

$$\begin{aligned} Adv_{\mathcal{D}}^{pw-mc}(B) &\leq \Pr[\boldsymbol{BAD}] + \Pr[Succ^{pw-mc} | \neg \boldsymbol{BAD}] \\ &\leq \Pr[\boldsymbol{BAD}] + q_s/N \end{aligned}$$

Next we consider the probability of the event **BAD**.

– If in the execution of the protocol, the client B does not modify X^*, then the client B also could compute $g^{xyrr'}$. If we treat rr' as a single random

number s, then what B could learn from the protocol is exactly the previous information given in PCDDH1 assumption. According to PCDDH1 assumption, **BAD**'s probability in one execution of the protocol is equal to the advance of client B in PCDDH1 assumption. Using standard hybrid proof techniques, we have

$$\Pr[\boldsymbol{BAD}] = q_s \cdot Adv_{\mathbb{G},N}^{PCDDH1}(B)$$

– If in the execution of the protocol the client B modifies X^*, we would bound **BAD**'s probability by the PCDDH2 assumption. Similarly we have

$$\Pr[\boldsymbol{BAD}] = q_s \cdot Adv_{\mathbb{G},N}^{PCDDH2}(B)$$

Hence,

$$Adv_{\mathcal{D}}^{pw-mc}(B) = q_s \cdot max\{Adv_{\mathbb{G},N}^{PCDDH1}(B), Adv_{\mathbb{G},N}^{PCDDH2}(B)\} + q_s/N \quad \square$$

Password Protection Against Malicious Server: In our protocol, we suppose S_2 is malicious and his goal is to learn client A's password pw_A. We also prove this security notion through PCDDH1 and PCDDH2 assumptions. But now the random number s used in the assumptions is defined as r rather than rr'. Following theorem's proof is similar as that of Theorem 2.

Theorem 3. *In our cross-realm C2C-PAKE protocol, the malicious server S_2 cannot learn the client A's password as long as the* **PCDDH1** *and* **PCDDH2** *assumptions hold in the group $\mathbb{G}$. Formally,*

$$Adv_{\mathcal{D}}^{pw-ms}(t, q_s) \leq q_s \cdot max\{Adv_{\mathbb{G},N}^{PCDDH1}(t), Adv_{\mathbb{G},N}^{PCDDH2}(t)\} + q_s/N$$

where q_s represents the number of queries to SendServer and SendClient.

5 Conclusion

From the introduction of cross-realm C2C-PAKE protocol, many implements have been proposed. But no formal consideration exists and no existing protocol could satisfy all security requirements. In fact, which security a cross-realm C2C-PAKE protocol should provide is never defined clearly. In this paper, we firstly consider security requirements about this special password-based authenticated key exchange protocol according to different adversaries: outside adversary, inside client and inside server. Formally, we define four security notions: (1) *semantic security against malicious outside adversary*, (2) *key privacy against passive server*, (3) *password protection against malicious client*, and (4) *password protection against malicious server*.

We design a new cross-realm C2C-PAKE protocol which is the first such protocol satisfying all four security notions. We also provide formal security proof under our model and security notions.

References

1. Michel Abdalla, Olivier Chevassut, and David Pointcheval. One-time verifier-based encrypted key exchange. In *PKC 2005*, LNCS 3386, pages 47–64. Springer, 2005.
2. Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In *PKC 2005*, LNCS 3386, pages 65–84. Springer, 2005.
3. Michel Abdalla and David Pointcheval. Interactive diffie-hellman assumptions with applications to password-based authentication. In *FC 2005*, LNCS 3570, pages 341–356. Springer, 2005.
4. M. Bellare and P. Rogaway. The autha protocol for password-based authenticated key exchange. In *Contribution to the IEEE P1363 study group*, 2000.
5. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, LNCS 1807, pages 139–155. Springer, 2000.
6. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO '93*, LNCS 773, pages 232–249. Springer, 1993.
7. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *STOC 1995*, pages 57–66. ACM, 1995.
8. Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *EUROCRYPT 2000*, LNCS 1807, pages 156–171. Springer, 2000.
9. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Security proofs for an efficient password-based key exchange. In *CCS 2003*, pages 241–250. ACM, 2003.
10. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. New security results on encrypted key exchange. In *PKC 2004*, LNCS 2947, pages 145–158. Springer, 2004.
11. Jin Wook Byun, Ik Rae Jeong, Dong Hoon Lee, and Chang-Seop Park. Password-authenticated key exchange between clients with different passwords. In *ICICS 2002*, LNCS 2513, pages 134–146. Springer, 2002.
12. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
13. Jeeyeon Kim, Seungjoo Kim, Jin Kwak, and Dongho Won. Cryptanalysis and improvement of password authenticated key exchange scheme between clients with different passwords. In *ICCSA 2004, Part I*, LNCS 3043, pages 895–902. Springer, 2004.
14. Philip D. MacKenzie. More efficient password-authenticated key exchange. In *CT-RSA 2001*, LNCS 2020, pages 361–377. Springer, 2001.
15. Philip D. MacKenzie. The pak suite: Protocols for password-authenticated key exchange. In *Submission to IEEE P1363.2*, 2002.
16. Raphael Chung-Wei Phan and Bok-Min Goi. Cryptanalysis of an improved client-to-client password-authenticated key exchange (c2c-pake) scheme. In *ACNS 2005*, LNCS 3531, pages 33–39, 2005.
17. Shuhong Wang, Jie Wang, and Maozhi Xu. Weaknesses of a password-authenticated key exchange protocol between clients with different passwords. In *ACNS 2004*, LNCS 3089, pages 414–425. Springer, 2004.

Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction

Praveen Gauravaram[1], William Millan[1], Ed Dawson[1], and Kapali Viswanathan[2]

[1] Information Security Institute (ISI)
Queensland University of Technology (QUT)
2 George Street, GPO Box 2434, Brisbane QLD 4001, Australia
p.gauravaram@isi.qut.edu.au, {b.millan, e.dawson}@qut.edu.au
[2] Technology Development Department, ABB Corporate Research Centre
ABB Global Services Limited, 49, Race Course Road, Bangalore - 560 001, India
kapaleeswaran.v@in.abb.com

Abstract. Recently multi-block collision attacks (MBCA) were found on the Merkle-Damgård (**MD**)-structure based hash functions MD5, SHA-0 and SHA-1. In this paper, we introduce a new cryptographic construction called **3C** devised by enhancing the **MD** construction. We show that the **3C** construction is at least as secure as the **MD** construction against single-block and multi-block collision attacks. This is the first result of this kind showing a generic construction which is at least as resistant as **MD** against MBCA. To further improve the resistance of the design against MBCA, we propose the **3C+** design as an enhancement of **3C**. Both these constructions are very simple adjustments to the **MD** construction and are immune to the straight forward extension attacks that apply to the **MD** hash function. We also show that **3C** resists some known generic attacks that work on the **MD** construction. Finally, we compare the security and efficiency features of **3C** with other **MD** based proposals.

Keywords: Merkle-Damgård construction, MBCA, 3C, 3C+.

1 Introduction

In 1989, Damgård [2] and Merkle [13] independently proposed a similar iterative structure to construct a collision resistant cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^t$ using a fixed length input collision resistant compression function $f : \{0,1\}^b \times \{0,1\}^t \rightarrow \{0,1\}^t$. Since then, this iterated design has been called Merkle-Damgård (**MD**) construction which influenced the designs of popular dedicated hash functions such as MD5, SHA-0 and SHA-1. The design motivation of the **MD** construction is that if the compression function f is collision resistant then so is the resultant iterated hash function H.

It is known that, a compression function f secure against the fixed initial value (IV) collisions is necessary but not sufficient to generate a secure hash function

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 407–420, 2006.
© Springer-Verlag Berlin Heidelberg 2006

H [12, p.373]. The latest multi-block collision attacks (MBCA) on the hash functions MD5, SHA-0 and SHA-1 [19, 1, 17, 18] prove this insufficiency. These attacks clearly show that these iterated hash functions do *not properly preserve* the collision resistance property of their respective compression functions with the fixed IV. The MBCA on hash functions leave open the questions; Is it possible to design collision resistant hash functions relying on the collision resistance of the compression function with fixed IV?, Is it possible to design a simple and efficient structures that offer more resistance to MBCA than the **MD** structure?

In this paper, we attempt to answer these questions. Our motivation is to show that while consecutive iterations of the compression function is necessary for the implementation efficiency of a hash function, **the way** the compression function is iterated or used is quite important for the security of the hash function. In this paper, we propose a new mode of operation for the **MD** construction called **3C**. The **3C** hash function processes the intermediate chaining values of the **MD** construction by maintaining a second internal chaining variable. The **3C** construction is the simplest modification of the **MD** construction that one can obtain to improve its security against MBCA.

We show that the **3C** construction is at least as secure as **MD** construction against collision attacks, in particular against MBCA. That is, if there exists an adversary that finds a multi-block collision on **3C** then that adversary would have also found a multi-block collision on the **MD** hash function. In addition, we show that if there exists an adversary that can perform an MBCA on a t-bit **MD** hash function based on a given compression function then the security of **3C** against MBCA instantiated with the same compression function could be as much as 2^t times the security of **MD** against MBCA, depending on the subtle properties of the compression function. We conjecture that this security multiplier of **3C** against MBCA is close to 2^t for any compression function which is secure against single-block collision attacks. We note that multiplier of at least $2^{t/2}$ is sufficient to provide immunity to MBCA. Next, extra memory is added to the **3C** construction and call this variant **3C+** and analyse the difficulty in implementing an MBCA on it compared to **3C**. Analysis for **3C** against known generic attacks [3, 8, 9] is given which applies to **3C+** as well. We found that while Joux's generic attacks [8] work on **3C**, the known generic second preimage attacks [3, 9] found on the **MD** hash function do not work.

In Section 2, we describe **MD** hashing and collision attacks on it. In Section 3, new observations on the MBCA are discussed. In section 4, **3C** is introduced and its analysis against MBCA is covered in Section 5. In Section 6, analysis of **3C** against generic attacks is given and **3C** is compared with some other similar hash function proposals in Section 7. In Section 8, **3C+** is introduced and is analysed against MBCA. The paper is concluded in Section 9.

2 MD Hashing and Collision Attacks

A collision resistant cryptographic hash function H following **MD** structure is a function that hashes a message $M \in \{0,1\}^*$ to outputs of fixed length $\{0,1\}^t$.

The specification of H includes the description of the compression function f, initial state value (IV) and a padding procedure [12, 14]. Every hash function fixes the IV (fixed IV) with an upper bound on the size $|M|$ of the input M. The message M is split into blocks $M_1, \ldots, M_{L-1}$ of equal length b where a block M_L containing the length $|M|$ (**MD** strengthening) [12] is added. Each block M_i is iterated using a fixed length input compression function f computing $H_i = f(H_{i-1}, M_i)$ where $i = 1$ to L and finally outputting $H_{IV}(M) = H_L$ as shown in Fig 1.

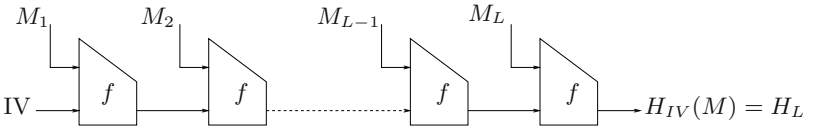


Fig. 1. The Merkle-Damgård (**MD**) construction

Collision Attacks on the Compression Functions
A hash function H is said to be collision resistant if it is hard to find any two distinct inputs M and N such that $H(M) = H(N)$. For a formal definition see [15]. A hash function H is said to be near-collision resistant if it is hard to find any two distinct inputs M and N such that $H(M) \oplus H(N) = \Delta$ has some small weight. Based on the IV used in finding collisions, collision attacks on the compression functions are classified as follows [12, p.372]:

1. Collision attack: collisions using a fixed IV for two distinct messages (e.g. [16]). We call them *Type 1* collisions.
2. Semi-free-start collision attack: collisions using the same random (or arbitrary) IV for two distinct message inputs(e.g. [5]). We call them *Type 2* collisions.
3. Pseudo-collision attack: free-start collision attack using two different IVs for two distinct message inputs(e.g. [4]). We call them *Type 3* collisions.

Multi-block Collision Attacks on Hash Functions
A multi-block collision attack (MBCA) technique on an iterated hash function finds two colliding messages each of at least two blocks in length. The recent collision attacks on MD5 [19], SHA-0 [1, 17] and SHA-1 [18] are multi-block collision attacks where collisions were found by processing more than one message block. Since, by far, most of the possible messages are more than a single block and collisions are distributed randomly, it is fair to say that most collisions that could exist are in fact multi-block collisions. Hence, any result improving the resistance against MBCA is very significant.

3 New Observations on Multi-block Collision Attacks

This section aims at developing the understanding of the multi-block collision attacks on hash functions by linking several parts of the literature. We observed

that multi-block collision attacks on hash functions can further be classified into three categories based on the manner in which the compression functions are attacked. This leads to choosing particular message block formats that result in an MBCA. For example, 2-block collision attacks can be classified into three types as shown in Table 1 based on the message formats chosen.

Table 1. Classification for 2-block collision attacks

MBCA Type	Message formats
MBCA-1	(M_1, M_2) and (M_1, N_2)
MBCA-2	(M_1, M_2) and (N_1, M_2)
MBCA-3	(M_1, M_2) and (N_1, N_2)

While the 2-block collision attacks on MD5 [19] and SHA-1 [18] belong to MBCA-3 category, the 2-block collision attack on SHA-0 [17] belongs to an MBCA-1 category[1]. In an MBCA-3, near-collisions found after processing first message blocks were converted to full collisions as was demonstrated on MD5 and SHA-1. The *Type 1* collisions were (reportedly) hard to find for the single compression functions of these hash algorithms. For example, the attacks on MD5 and SHA-1 use near-collisions obtained after processing the first distinct message blocks (M_1, N_1) as a tool to find collisions for the second distinct message blocks (M_2, N_2). This technique can be generalized to more than two blocks as the 4-block collision attack on SHA-0 [1]. Similarly, 2-block MBCA-1 and MBCA-2 attack techniques can be generalized to more than two blocks. For example, in an MBCA-1 technique [17], a few initial message blocks to be processed can be chosen to be the same to satisfy certain conditions required in the attack followed by the processing of two different message blocks that give a collision.

We note that in a 2-block MBCA, collisions found on the second blocks are basically a *special case* of *Type-3* collisions for the compression function as these collisions require processing of two equal (MBCA-1) or different blocks (MBCA-2 and MBCA-3) using the fixed IV of the hash function. That is, a 2-block MBCA-3 on the **MD** hash function is a combination of a near-collision and a *special Type-3* on the compression function. The collision on the second compression function is a *special Type-3* as it requires a particular nearly collided value obtained based on certain conditions essential for the attack as inputs for the second block messages. In addition, near-collisions do not need to begin after processing message blocks based on the fixed IV of the hash function. They can also be due to an arbitrary chaining value when the attacker chooses the same blocks initially and starts an MBCA-3 after processing those same initial blocks. Hence multi-block difference collisions, whether they start from the fixed IV or arbitrary chaining values are clearly a chain of *special Type 3* collisions.

[1] The first full 4-block collision attack on SHA-0 [1] also belongs to an MBCA-3 category except that differences in the message blocks span over more than two blocks.

Table 2. Resistances of some compression functions

Compression function	*Type-1*	*Type-2*	*Type-3*	*Special Type-3*
MD4	NO [16]	NO	NO	-
MD5	YES	NO [5]	NO [4]	NO [19]
SHA-0	YES	NO [19]	YES	NO [1]
SHA-1	YES	YES	YES	NO [18]
RIPEMD	NO [16]	NO	NO	-
HAVAL-128	NO [16]	NO	NO	-

From these observations on MBCA, it is clear that the designers of MD5, SHA-0 and SHA-1 *have not considered security of the compression functions of these hash functions against special Type-3 collisions in their design criteria.* Preneel pointed out more than a decade back [14] that most hash functions are not designed to meet this criteria. Note that SHA-1 did not exist then. Even Damgård's [2] proof implicitly notes that the necessity of *special Type-3* collision resistance for the compression functions. In addition, to attain *Type-3* collisions, the two IVs do not have to be significantly different as suggested in [12, p.372]. For example, the two IVs in the *Type-3* collision attack on the compression function of MD5 [4] differ in *only* 6 bits. From the known attacks on hash functions, we derived Table 2 assuming that if the compression function is not *Type-1* collision resistant then it is neither *Type-2* nor *Type-3* collision resistant. The sign "-" in the Table 2 shows does not apply.

4 The 3C Construction: An Enhanced MD Construction

The **3C** construction is shown in Fig. 2 and 3. This structure has an *accumulator* XOR function iterated in the *accumulation chain* (whose chaining value is denoted by u_i in Fig. 3) and a compression function f (f, for example, is the compression function of MD5 or SHA-1) iterated in the *cascade chain* (whose chaining value is denoted by w_i in Fig. 3) exactly as in the **MD** construction. Clearly, **3C** is a very simple and efficient modification to the **MD** construction. One economic benefit of our proposal is that any software currently implementing an **MD**-style hash function can be very simply altered to adopt the **3C** structure, without altering the underlying compression function.

3C hashing process: For $i = 1$ to L, let w_i and u_i be the chaining values in the *cascade chain* and *accumulation chain* respectively. Then, as in the **MD** hash, for $i = 1$ to L, $w_i = f(w_{i-1}, M_i)$ where $w_0 = IV$ and $u_1 = w_1$. In the *accumulation chain*, for $i = 2$ to L, $u_i = u_{i-1} \oplus w_i$. The result u_L in the *accumulation chain* is denoted with Z. An extra compression function f, denoted by g, is added at the end and the hash result of **3C** is $g(\overline{Z}, w_L)$. To process one block data, the compression function is executed three times; first to process the data block, next to process the padded block (**MD** strengthening) and finally the block $\overline{Z}$ formed in the *accumulation chain* as shown in Fig 3. If the size of data is less than block size b of f then zeros are appended to the data to make a b-bit data block.

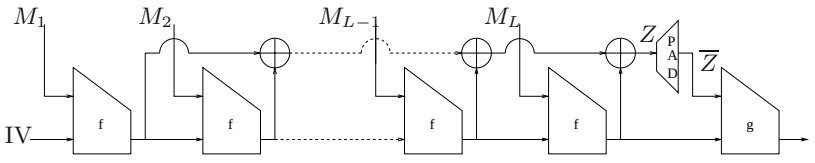


Fig. 2. The **3C**-hash function

5 Security Analysis of the 3C Hash Function

In this section, we investigate the security of **3C** against single-block and multi-block collision attacks. We conclude that the security of **3C** against single-block collision attacks is upper bounded by the collision security of the compression function and its security against MBCA is not less than that on **MD**. Fig 3 is used to explain the analysis.

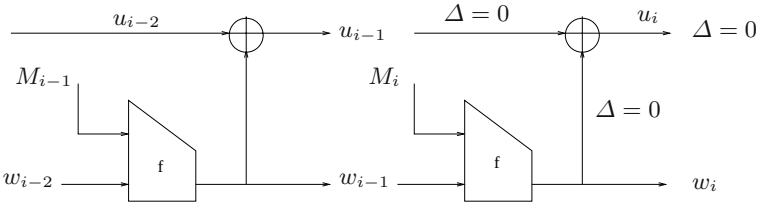


Fig. 3. Creating an internal collision for **3C**

Consider a **3C** hash function H. Consider two distinct messages $M \neq N$ of same length L (including padding) such that $H(M) = H(N)$ is the result of a collision on **3C**. The messages M and N are expanded to sequences $(M_1, \ldots, M_L) \neq (N_1, \ldots, N_L)$ where the last data blocks are the padded blocks containing the length L of the messages. We denote by (H_i^M, H_i^N) and (u_i, v_i) (for $i = 1$ to L), the internal hash values obtained on the *cascade chain* and *accumulation chain* while computing $H(M)$ and $H(N)$ respectively. We denote (u_L, v_L) by (Z_M, Z_N) and $\overline{Z}_M = \text{PAD}(Z_M)$, $\overline{Z}_N = \text{PAD}(Z_N)$. All possible types of collisions on H are given in Definition 1.

Definition 1. *Every collision on H takes one of the following forms:*

1. *Terminal/Final collisions on H: They involve one of the following cases:*
   - $H_L^M \neq H_L^N$ *and* $\overline{Z}_M \neq \overline{Z}_N$ *with* $g(H_L^M, \overline{Z}_M) = g(H_L^N, \overline{Z}_N)$
   - $H_L^M = H_L^N$ *and* $\overline{Z}_M \neq \overline{Z}_N$ *with* $g(H_L^M, \overline{Z}_M) = g(H_L^N, \overline{Z}_N)$
   - $H_L^M \neq H_L^N$ *and* $\overline{Z}_M = \overline{Z}_N$ *with* $g(H_L^M, \overline{Z}_M) = g(H_L^N, \overline{Z}_N)$
2. *Internal collisions on H: $H_L^M = H_L^N$ and $\overline{Z}_M = \overline{Z}_N$ implies $g(H_L^M, \overline{Z}_M) = g(H_L^N, \overline{Z}_N)$.* □

Definition 2. *A compression function $f : \{0,1\}^b \rightarrow \{0,1\}^t$ is Type-1* (resp. *Type-2, Type-3*) *collision resistant if the best possible collision attack on it using fixed IV* (resp. *arbitrary IV, different IVs*) *is the birthday attack which takes about $2^{t/2}$ operations of f. For sufficiently large t, it is computationally infeasible to perform this attack.*

Lemma 1. *Against single block collision attacks, the security of* **3C** *is exactly equal to the security of* **MD** *when both the constructions are instantiated with the same f.*

Proof: By inspection of **3C** structure in Fig 3, it is clear that it contains **MD** construction in it. Hence, an adversary that is able to find a single block collision for the f-function is able to construct a collision for the hash function with virtually no additional effort. It follows that the collision security of **3C** is upper bounded by the collision security of the f-function. □

Apart from single block collision attacks, the only other approach to find collisions for **3C** is to use multi-block messages. This invites the opportunity to use messages with different lengths. For two messages with the same length, an internal collision for **3C** gives an actual collision for **3C**. However, for two messages of different lengths, in general, this is not the case due to the different padding strings used as a virtual message block in the second last iteration of the compression function. Thus the security analysis of **3C** can be restricted to considering internal collisions generated by pairs of messages with the same length. We now examine the nature of internal collisions for **3C**.

Lemma 2. *To get an internal collision on* **3C** *at iteration i, it is required that a collision in the accumulation chain exists at iteration $i-1$.*

Proof: An internal collision in **3C** at iteration i is a simultaneous collision in the *accumulation chain* and *cascade chain* at iteration i. For messages M and N to collide on the *cascade chain* at iteration i, the condition that $H_i^M \oplus H_i^N = 0$ must be satisfied. Now for an internal collision on **3C**, the condition that $(H_i^M \oplus H_i^N) \oplus (u_{i-1} \oplus v_{i-1}) = 0$ must be satisfied. This condition will occur only when $u_{i-1} \oplus v_{i-1} = 0$, which is basically a collision in the *accumulation chain* at iteration $i-1$. □

Remark: A collision in the accumulation chain at iteration $i-1$ is achieved by creating a sequence of **MD** chain differences where the chaining difference in the **MD** chain at iteration $i-1$ is the XOR sum of all the previous differences in the **MD** chain until the iteration $i-2$.

Lemma 3. *Assuming the existence of a collision in the accumulation chain at iteration $i-1$, it requires a single-block special Type-3 collision attack on f to create an internal collision in* **3C** *at iteration i.*

Proof: By inspection of **3C** structure in Fig 3, a special Type-3 collision attack must be performed on the f-function at iteration i. It is a *special Type-3* collision

attack as the attacker must use the internal chaining values of the *cascade chain* at iteration $i-1$ that created a collision in the *accumulation chain* as inputs to get a collision at iteration i. This is equivalent to performing a single-block *special Type-3* collision attack on f at iteration i. □

Now we can consider the above process in two ways: as two separate single block collision attacks, or as a multi-block collision attack on the MD-chain with an extra t-bit requirement. We ignore the first option as the single block collision security of the f-function is already an upper bound for the collision security of 3C from Lemma 1. The second case can be achieved in either of the following two ways:

1. Assume the existence of an MBCA on the **MD**-chain when it is instantiated with some given compression function. Then the MBCA on the **MD**-chain has to be repeated until the internal chaining differences on the **MD**-chain happen to produce the required collision on both the accumulation and cascade chains. This option requires repeating the attack at most 2^t times under an assumption that the internal chaining differences are uniformly distributed.
2. Devise an entirely new MBCA for the **3C** when instantiated with some given compression function satisfying some conditions on the differences.

Now, the above two cases result in the following theorems:

Theorem 1. *If there is an MBCA on the* **MD** *construction instantiated with a given f then the security of* **3C** *instantiated with the same f against an MBCA is at most 2^t times the security of* **MD** *against MBCA.*

Proof: To obtain a collision in the *accumulation chain* required by Lemma 2, an MBCA on the **MD** chain must be repeated, where each attempt succeeds with probability 2^{-t}. That is, the security of **3C** against MBCA is some multiple ($[1, 2^t]$) of the security against MBCA for the **MD**. □

The difficulty of providing a tight quantitative analysis for **3C** against MBCA prevents a more precise formal proof for the practical collision security of **3C** at this stage leading to the following conjecture.

Conjecture: From the above analysis, we conjecture that the improvement in the security of **3C** against MBCA is close to 2^t over the security of **MD** against MBCA.

Theorem 2. *The security of* **3C** *against an MBCA is not less than the security of* **MD** *against MBCA.*

Proof: Every internal collision for **3C** contains within it a collision for **MD**. There exist collisions for **MD** that are not internal collisions for **3C**. Thus the security of **3C** against MBCA is lower bounded by the security of **MD** against MBCA. □

Remarks: While at least two blocks must be processed to find a multi-block collision on **MD**, at least three blocks must be processed to create a multi-block

collision on **3C**. For example, the difference pattern $(0, \Delta, \Delta, 0)$ which creates a collision on **MD** based on a given f, will also create a collision on **3C** based on the same f. But note that its reduced pattern $(0, \Delta)$ would create a collision for **MD** but not for **3C**. In Section 8, we propose a construction called **3C+** with similar properties to **3C** as an improvement of **3C** for more protection against MBCA.

6 Security Analysis of 3C Against Known Generic Attacks

Analysis against Joux attacks

Joux [8] described a generic multicollision attack on the **MD** hash where constructing 2^d-collisions costs d times as much as building ordinary 2-collisions. This attack can be used as a tool to find multi (2^{nd}) preimages very effectively on the **MD** hash. We note that these attacks work on **3C** as effectively as they are on the **MD** hash. Following [10], our adversaries are probabilistic algorithms and we focus on the expected running time. Running time is described asymptotically. We use the symbol O for the "expected running time is asymptotically at most".

In a multicollision attack on **3C**, the attacker finds collisions on every function f in the *cascade chain* (for example using the birthday attack) that would result in a collision at the subsequent point of the XOR operation in the *accumulation chain*. If the function f in the *cascade chain* of **3C** is modeled as a random oracle, as an upper bound, the total complexity to find 2^d-collisions on **3C** is $O(d * 2^{t/2})$.

We note that the attack technique used to find D-way (2^{nd}) preimages on the **MD** hash for a given hash value works on **3C** as well. For example on **3C**, the attacker first finds D-collisions on d-block messages $M^1, \ldots, M^{2^d}$ with $H_d = H(M^1) = \ldots = H(M^{2^d})$ with a complexity of $O(d * 2^{t/2})$. Then she finds the block M_{d+1} such that the execution of the last two compression functions would result in the given digest Y. The later task takes time $O(2^t)$ as the last two compression functions are treated as a single component. Hence the total cost of finding D-preimages for **3C** is $O(d * 2^{t/2} + 2^t)$. To find D-2^{nd} preimages for a given message M, the attacker first computes the hash $H(M)$ of the message M and then finds D-preimages as explained above that all collide to $H(M)$.

Analysis against second-preimage attacks

Dean [3] has demonstrated that for hash functions with fixed point compression functions, it would cost less than 2^t effort to find second preimages. Kelsey and Schneier [9] have expanded this result using Joux multicollision finding a technique to find second preimages for hash functions based on any compression function for an effort less than 2^t. Both these attacks use the notion of *expandable messages*- patterns of messages of different lengths that all process to internal hash values without considering **MD** strengthening. Following [9], an (a, b)-expandable message will take on any length between a and b message blocks.

For a compression function $H_i = f(H_{i-1}, M_i)$, a fixed point is a pair (H_{i-1}, M_i) such that $H_{i-1} = f(H_{i-1}, M_i)$. The compression functions of many hash functions such as MD5 and SHA-1 are Davies-Meyer designs with a block cipher operating in a feed-forward mode. For these compression functions, there exists one and only one fixed point for every message block. For a t-bit hash function with a maximum of 2^d blocks in its messages, using fixed points it costs about $2^{t/2+1}$ compression function computations to find $(1, 2^d)$-expandable messages [9]. In the **3C** design, since the chaining state is twice as large as the hash value, a fixed point is defined for both the chains and this is obtained for any message block M_i, *only when* $f(0, M_i) = 0$ and this occurs with a probability of 2^{-t}. Hence having fixed points for the compression functions will not assist in finding second preimages for less than 2^t work on the **3C** design.

It was demonstrated in [9] that finding a $(d, d + 2^d - 1)$ expandable message for any compression function with t-bit state takes only $d \times 2^{t/2+1}$ effort. The procedure involves first finding colliding pair of messages, one of one block and the other of $2^{d-1} + 1$ blocks starting from the initial state of the hash function. Then using the collided state as the starting state, collision pair of length either 1 or $2^{d-2}+1$ is found and this process is continued until a collision pair of length 1 or 2 is reached. It was shown in [9] that applying this generic expandable message finding algorithm to find the second preimage for a message of $2^d + d + 1$ -block length message costs $d \times 2^{t/2+1} + 2^{n-d+1}$ compression function computations. When this attack technique is applied on **3C**, a collision at both the chains is required and this costs an effort of 2^t at every stage as the size of the internal state is twice that of the hash size.But if different parts of the internal state of **3C** are attacked separately, **3C** might not resist the second preimage attack.

7 Comparison of 3C with Other Hash Function Proposals

Ferguson and Schneier [6] proposed double-hashing scheme $H_{IV}(H_{IV}(x))$ to prevent straight-forward length extension attacks. It is obvious that multi block collision attacks work on this nested construction as effectively as they are on **MD**. As on the **MD** hash, 2^d-collisions can be found on their scheme with a complexity of $O(d.2^{t/2})$ and finding 2^d-(2^{nd}) preimages would take time $O(d.2^{t/2} + 2^t)$. Gauravaram *et al.* [7] proposed CRUSH hash function based on iterated length halving technique as an alternative for **MD** hash function well before the invention of MBCA on MD5 anticipating the single point of failure of hash functions in the MD family. CRUSH is immune against extension attacks and resists known MBCA techniques.

Lucks [10] proposed wide-pipe and its special case double-pipe hash designs as failure-tolerant designs showing that they provide more resistance against generic attacks [8] than the **MD** hash. While wide-pipe hash maintains more internal state than the hash size t using larger compression functions, double-pipe hash maintains twice the hash size as the internal state size by employing one single t-bit compression function used twice in parallel for each message block. In contrast, one could see **3C** structure as special cases of wide-pipe hash and

is optimally efficient as no new large compression function needs to be designed for its execution. The wide-pipe, **3C** and double-hashing proposals resist the straight-forward length extension attacks which is a well-known weakness of the **MD** hash function. Informally, given the digest H of the message M, it is straight forward to compute N and H' such that $H' = H(M||N)$ even for unknown M but for known $|M|$. The attack uses $H(M)$ as the internal hash value to compute $H(M||N)$. All these hash functions provide $t/2$-bit level of security against straight forward extension attacks as long as their design criteria is satisfied; for example, wide-pipe hash requires processing of the compression function with an internal state at least twice the size of the hash value, **3C** requires at least three calls to the compression function. Note that **3C** prevents extension attacks with out using large compression function as in the wide-pipe hash.

While the wide-pipe and double-pipe hash functions are designed to provide more resistance against generic attacks, **3C** and **3C+** are enhancements of **MD** resisting recent multi-block collision attacks on the **MD** based hash functions. In addition, one can combine the wide-pipe hash and the **3C** construction to attain a hybrid construction called 3CWP (see Fig 4) attaining additional protection against both the generic attacks and MBCA.

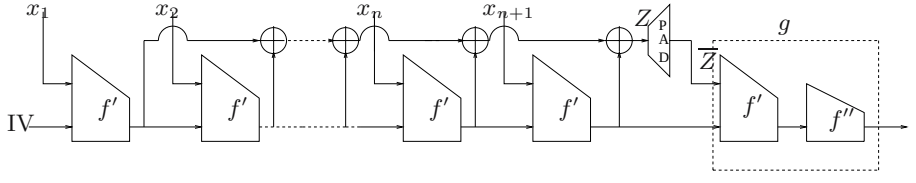


Fig. 4. The **3CWP** hybrid construction

From the performance point of view, **3C** is slightly more expensive than **MD** especially when it is used to process short messages as the former requires at least three iterations of the compression function to process an arbitrary length message. To process 1-block (resp. 2-block) message, the running time of the **3C** is twice (resp. 1.5 times) that of **MD**. On an Intel Pentium 4 3.2GHz processor, **3C** based on the compression of MD5, incurs about 0.36% overhead and **3C** with the compression function of SHA-1 incurs about 0.27% overhead when these functions are used to process long messages. **3C** requires an extra iteration of the compression function similar to the double hashing proposal [6] and is as efficient as this scheme for the processing of long messages and unlike the double hashing scheme, **3C** is a single hashing scheme.

8 The 3C+ Construction: An Enhanced 3C Construction

Fig 5 shows the **3C+** construction where a third internal chain called *final chain* has been added on top of the *cascade* and *accumulation chains* of **3C**. In **3C+**,

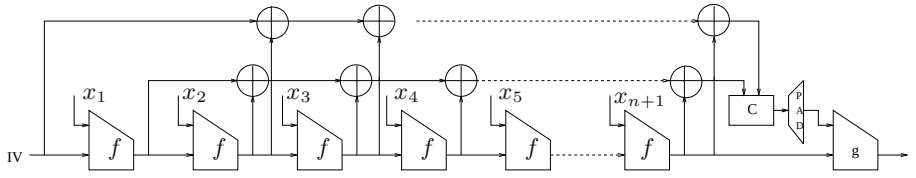


Fig. 5. The **3C+** hash construction

we call *accumulation chain* as the *middle chain*. The *final chain* in **3C+** slightly differs in the way it accumulates data from the *accumulation chain* of **3C** as it accumulates data from the *cascade chain* but the accumulation starts after processing the second message block. The final compression function f (denoted by g in Fig 5) takes as "message" the concatenation of the accumulated data from the *middle* and *final* chains, appropriately padded.

To find a multi-block collision on **3C+** the attacker has to get collisions simultaneously on all the three chains. To create a simultaneous collision on all the three chains at iteration i, the chaining difference on the *cascade chain* at iteration $i-1$ (say Δ_{i-1}) must cancel the differences accumulated in the *middle* and *final* chains till the iteration $i-2$. That is, the *middle* and *final* chains must maintain an equal difference Δ_{i-1} until the iteration $i-2$ of the function f for a cancellation at iteration $i-1$. This is impossible if *middle* and *final* chains do not start with the same difference before iteration $i-2$. That is, if an attacker finds two colliding messages that have identical first message blocks, then these two chains begin with a difference zero. This implies that a minimum of four message blocks need to be processed to find an MBCA on **3C+**. For example, the pattern $(0, 0, \Delta, \Delta, 0)$ creates a simultaneous collision on all the chains after processing four blocks. From this discussion, it is clear that **3C+** structure demands crafting and maintaining the same difference in both the *final* and *middle* chains until a multi-block collision is found.

Finally, we note that if the input to the *final chain* is taken from the *middle chain* rather than from the *cascade chain*, the difference pattern $(0, \Delta, \Delta, 0)$ that creates a collision on **MD** and **3C** will also create a collision on this modified construction of **3C+** and the MBCA security of this construction is lower bounded by **3C** and **MD**. Note this pattern does not create a collision on **3C+**. This justifies our statement that the security of a hash function depends on the **way** the compression function is used in constructing a hash function.

We note that one can construct many variants for our **3C** and **3C+** designs by replacing XOR functions with any function in such a way that the new construction is at least as secure as **MD**. Here we provide two examples. If the XOR function in **3C** is replaced with the function f then this modified construction resembles double pipe hash in some way and is less efficient than **3C** against MBCA. The chaining values on the *cascade chain* after the second iteration of f will be used as data block (by appending 0's to chaining values) inputs for the compression functions in the accumulation chain. Clearly the amount of control that an attacker can have on these blocks to create an MBCA is less than **3C**.

A slight variant of **3C+** whose cost relative to **3C** bound to be nearly as small as XOR can be designed by interpreting the t-bit chaining value in the *final chain* as an element of GF(2^t) and multiplying it by 2 at each step. If the *final chain* accumulation process starts after the first iteration of f unlike in **3C+** then this structure results in a *final chain* accumulation equation that resembles Galois-Carter-Wegman structure of GHASH in [11].

9 Conclusion

The recent cryptanalysis of hash functions MD5, SHA-0, SHA-1 exploited the **MD** iterative structure of these hash functions using multi-block collision search techniques. The proposed **3C** and **3C+** variants to the **MD** construction are at least as resistant as **MD** against MBCA. The constructions can be implemented by simple adjustments to the existing **MD**-style implementations. This paper is the first paper to introduce solutions to MBCA on **MD** based hash functions since they were first identified by Wang *et al* on MD5 [16]. This paper will not and should not be taken as the last step but should be regarded by the Crypto community as the first step to improve the general design of hash functions.

Acknowledgments

Thanks to anonymous reviewers of ACISP 2006 for many useful comments on several aspects of the paper and their valuable insights. Many thanks to Suganya Annadurai, Paulo Barreto, Matt Henricksen, John Kelsey, Lars Knudsen, Adrian McCullagh, David McGrew, Juanma González Nieto, Vincent Rijmen and Søren Thomsen for their encouragement and comments on the analysis, design and performance aspects presented in the earlier drafts.

References

1. Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 36–57. Springer, 2005.
2. Ivan Damgard. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology: CRYPTO 89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1989.
3. Richard Drews Dean. *Formal Aspects of Mobile Code Security*. PhD thesis, Princeton University, 1999.
4. Bert denBoer and Antoon Bosselaers. Collisions for the compression function of MD5. In T. Helleseth, editor, *Advances in Cryptology — Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 293–304, Berlin, 1994. Springer-Verlag.
5. Hans Dobbertin. Cryptanalysis of MD5 compress. Presented at the rump session of Euro Crypto'96 Rump Session, 1996.

6. Niels Ferguson and Bruce Schneier. *Practical Cryptography*, chapter Hash Functions, pages 83–96. John Wiley & Sons, 2003.
7. Praveen Gauravaram, William Millan, and Lauren May. CRUSH: A New Cryptographic Hash Function using Iterated Halving Technique. In *Proceedings of the workshop on Cryptographic Algorithms and their uses*, pages 28–39, Goldcoast, Australia, July 4–5 2004.
8. Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Matt Franklin, editor, *Advances in Cryptology-CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316, Santa Barbara, California, USA, August 15–19 2004. Springer.
9. John Kelsey and Bruce Schneier. Second Preimages on n-bit Hash Functions for Much Less than 2n̂ Work. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer, 2005.
10. Stefan Lucks. A Failure-Friendly Design Principle for Hash Functions. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494. Springer-Verlag, 2005.
11. David McGrew and John Viega. The Galois/Counter Mode of Operation (gcm). NIST special publication, National Institute for Standards and Technology.
12. Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*, chapter Hash Functions and Data Integrity, pages 321–383. The CRC Press series on discrete mathematics and its applications. CRC Press, 1997.
13. Ralph 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, 1989.
14. Bart Preneel. *Analysis and design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
15. Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption (FSE)*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer-Verlag, 2004.
16. Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, 2004. `http://eprint.iacr.org/`.
17. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Efficient collision search attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology—CRYPTO '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005, 14–18 August 2005.
18. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology—CRYPTO '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005, 14–18 August 2005.
19. 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. Springer, 2005.

Forgery and Key Recovery Attacks on PMAC and Mitchell's TMAC Variant

Changhoon Lee[1], Jongsung Kim[1,2], Jaechul Sung[3], Seokhie Hong[1], and Sangjin Lee[1]

[1] Center for Information Security Technologies(CIST), Korea University, Anam Dong, Sungbuk Gu, Seoul, Korea
{crypto77, joshep, 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. In this paper we discuss the security of PMAC, a provably secure and parallelizable MAC scheme proposed by Black and Rogaway, and Michell's TMAC variant, proposed to improve the security of TMAC. We show how to devise forgery attacks on PMAC and compare the success rate of our forgery attacks with their security bound. We also present forgery attacks on TMAC variant and show the security of TMAC variant is not improved in the sense of the forgery attack. Furthermore, key recovery attacks on PMAC and TMAC variant are presented in various parameters. Our results imply they have no significant advantage in comparison with other well-established MAC schemes.

Keywords: MAC (Message Authentication Code), Forgery Attacks, Key Recovery Attacks, CBC-MAC, PMAC, TMAC Variant.

1 Introduction

A Message Authentication Code (MAC) scheme is a symmetric-key cryptosystem that is extensively used to protect a message from unauthorized alteration in internet security protocols and in banking applications. Generally, a MAC scheme consists of a tag generation algorithm (for a sender) and a tag verification algorithm (for a receiver). The tag generation algorithm takes as input a message M, a secret key K and a nonce IV, and returns a tag T. The verification algorithm takes as input M, T, K and IV, and returns a binary value indication whether or not the message/tag/nonce tuple is valid.

In the standard model of evaluating MAC security, there are two main kinds of practical attacks on MAC algorithms. One is a forgery attack in which an attacker tries to produce a message with a valid MAC without the knowledge of the secret key K. The other is a key recovery attack in which an attacker tries to obtain the secret key used to generate MAC values. A successful key recovery attack also enables the construction of arbitrary numbers of forgeries.

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 421–431, 2006.
© Springer-Verlag Berlin Heidelberg 2006

PMAC [2], which uses a k-bit key and an arbitrary-bit length message, and outputs a τ-bit tag, was proposed as a fully parallelizable alternative to CBC-MAC. It is provably secure under the assumption that the underlying n-bit block cipher is a pseudorandom permutation. More precisely, the upper bound on the advantage of a forgery attacker for PMAC is $\frac{(\sigma+1)^2}{2^{n-1}}$, where $\sigma = \sum_{i=1}^{q} \lceil \frac{|M_i|}{n} \rceil$ and M_i are message queries. However, there is no known cryptanalytic result on PMAC, which quantifies the number of message queries with which the forgery attacker can break PMAC.

TMAC [11], which requires two keys, a k-bit key and a n-bit key, was proposed by Kurosawa and Iwata. It is a refinement of XCBC-MAC [4] with three different keys and attains a provable security as the EMAC [19] does. Recently, some researchers [21, 15, 8, 16] evaluated the security of TMAC. Among them, Mitchell [15, 16] presented various attacks on TMAC and proposed a new TMAC variant to improve the security of TMAC.

In this paper we discuss the security of PMAC and TMAC variant against the forgery and the key recovery attacks. We first show forgery attacks on PMAC, which provide a lower bound on the attacker's advantage together with a required number of message queries. Our attack on PMAC where truncation is performed requires about $2^{\frac{n}{2}+1}$ known messages and $2^{n-\tau}$ MAC verifications where τ is the bit-size of the MAC, and the attack on PMAC where no truncation is performed requires about $2^{\frac{n}{2}+1}$ known messages. All these attacks work with a success rate of 0.63. The latter attack shows that there exists a forgery attack on PMAC with $2^{\frac{n}{2}+1}$ message queries (all are one-block messages) such that the lower bound on the advantage of the attacker for PMAC is $L_b = \frac{(\sigma+1)^2}{2^{n+3}}$, since $\sigma = 2^{\frac{n}{2}+1}$ implies $L_b \approx 0.5 (< 0.63)$ (more details will be given in Sect. 3.1). Note that it does not contradict the security proof of PMAC in [2] and $L_b = \frac{1}{16} \cdot U_b$ for the upper bound $U_b = \frac{(\sigma+1)^2}{2^{n-1}}$. We also give a forgery attack on the TMAC variant whose complexity is equal to that of the forgery attack on TMAC, $2^{\frac{n}{2}+1}$ known messages. This fact implies that the security of TMAC variant is not improved in the sense of the forgery attack. Furthermore, key recovery attacks on PMAC and TMAC variant are also presented in various parameters.

This paper is organized as follows: The descriptions of PMAC, TMAC variant are presented in Sect. 2. In Sect. 3 and Sect. 4, forgery and key recovery attacks on PMAC and TMAC variant are presented, respectively. Finally, we summarize our attacks and compare them with previous attacks on other MAC schemes in Sect. 5.

2 Descriptions of PMAC and TMAC Variant

The most common block-cipher based MAC is CBC-MAC [5]. This scheme operates as follows: Suppose the underlying block cipher E has an n-bit block and uses a k-bit key K. We write $E_K(x)$ for the encryption of x using key K, where x is an n-bit block. A padded message M is first split into $M[1], ..., M[r]$ where each block $M[i]$ is of size n-bit and then the CBC-MAC value of the M is computed as CBC-MAC$_K(M) = H_r$, where $H_0 = 0$, $H_i = E_K(H_{i-1} \oplus M[i])$, $1 \le i \le r$.

2.1 PMAC (Parallelizable Message Authentication Code)

Unlike popular MAC algorithms such as CBC-MAC and Hashed MAC, PMAC proposed by Black and Rogaway is not inherently sequential but parallelizable. It also uses a key K and an additional key information $L = E_K(0^n)$ which is defined from K. Furthermore, it can handle messages which are not multiples of the block length without the need for obligatory padding, which would increase the number of block cipher calls. PMAC operates as follows (See Fig. 1):

Algorithm $\mathrm{PMAC}_K(M)$
1. $L \leftarrow E_K(0^n)$
2. if $|M| > n2^n$ then return 0^τ
3. Partition M into $M[1] \cdots M[r]$
4. for $i \leftarrow 1$ to $r-1$ do
5. $\quad X[i] \leftarrow M[i] \oplus \gamma_i \cdot L$
6. $\quad Y[i] \leftarrow E_K(X[i])$
7. $\Sigma \leftarrow Y[1] \oplus \cdots \oplus Y[r-1] \oplus pad(M[r])$
8. if $|M[r]| = n$ then $X[r] = \Sigma \oplus L \cdot x^{-1}$ else $X[r] \leftarrow \Sigma$
9. $MAC = truncation\ of\ first\ \tau\ bits\ of\ (E_K(X[r]))$
10. return MAC

PMAC: Constants γ_i are Gray codes which are polynomial multiplication operators over $GF(2^n)$. The $pad(A)$ means the string $A||10^{n-|A|-1}$ in the case $|A| < n$.

2.2 TMAC Variant

TMAC [11], which requires two keys, a k-bit K and an n-bit K', was proposed by Kurosawa and Iwata. It is a refinement of XCBC [4] using $(k+2n)$-bit keys, that is, a key triple (K_1, K_2, K_3) used in XCBC is replaced by $(K, K' \cdot u, K')$, where u is a constant which is a multiplication operator over $GF(2^n)$ defined in [11]. The usage of the $(K, K' \cdot u, K')$ depends on whether or not padding has been necessary. If the length of the input message M is a multiple of n, then TMAC operates exactly the same as CBC-MAC using the key K, except for XORing an n-bit key $K' \cdot u$ before encrypting the last block. Otherwise, after padding is applied, TMAC operates CBC-MAC using the key K, but this time an n-bit key K' is XORed before the last block is encrypted.

Recently, several attacks on TMAC was presented in [21, 15, 8, 16]. In particular, Mitchell [15, 16] considered various attacks (including forgery attacks, partial key attacks and key recovery attacks) against TMAC and proposed a new TMAC variant to remove a simple algebraic relationship between $K' \cdot u$ and K' which is used as the basis of the key recovery attacks. The only difference between this variant and TMAC is to use a key triple, i.e., $(K, E_{K'}(S_2), E_{K'}(S_3))$ substitutes for $(K, K' \cdot u, K')$ in the TMAC variant, where S_2 and S_3 are different fixed n-bit strings. For convenience we call this scheme TMAC-V. Fig. 2 represents the TMAC-V scheme.

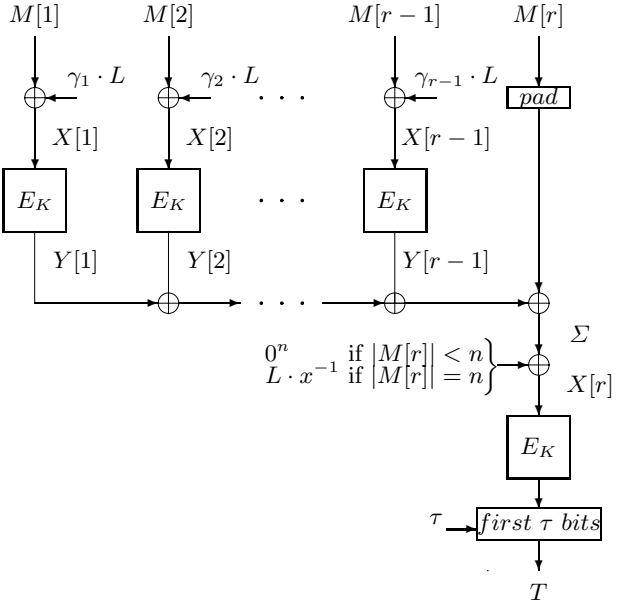


Fig. 1. The PMAC scheme

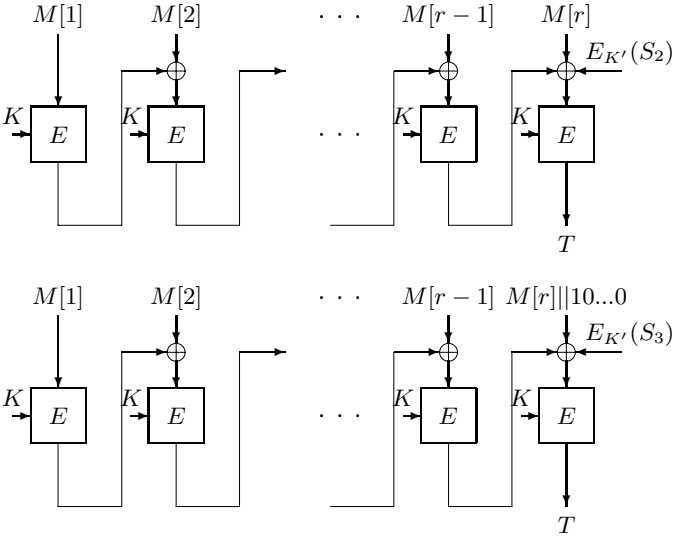


Fig. 2. The TMAC-V scheme

3 Forgery and Key Recovery Attacks on PMAC

In this section we present forgery and key recovery attacks on PMAC where truncation is used or not.

3.1 Forgery Attacks on PMAC

For clarification we separately describe our forgery attacks on PMAC with no truncation and on PMAC with truncation.

PMAC where no truncation is performed ($\tau = n$): We first consider the case where no truncation is performed. Assume that the attacker obtains the corresponding MAC values for approximately $2^{n/2}$ different $(q+1)$-block messages $M(1)^i = (M[1], ..., M[q], X^i)$ where $(M[1], ..., M[q])$ is any fixed sequence of n-bit blocks and the last blocks $X^i (1 \leq i \leq 2^{n/2})$ whose sizes are less than n bits are pairwise-distinct, and q is an arbitrary positive integer. Note that all these messages require padding. Assume that the attacker also obtains the corresponding MAC values for approximately $2^{n/2}$ different $(q+1)$-block messages $M(2)^j = (M[1], ..., M[q], Z^j)$ where $(M[1], ..., M[q])$ is the same sequence as that in $M(1)^i$ and the last blocks $Z^j (1 \leq j \leq 2^{n/2})$ whose sizes are equal to n bits are pairwise-distinct.

Using the birthday paradox arguments [17] we can expect to find at least one external collision with a probability of $0.63 (\approx 1 - e^{-1})$. In other words, one of the MAC values from the first set of messages $M(1)^i$ will equal to one of the MAC values from the second set of messages $M(2)^j$ with a probability of 0.63. Denote by $M(1)^* = (M[1], ..., M[q], X^*)$ and $M(2)^* = (M[1], ..., M[q], Z^*)$ the pair of messages which cause this collision. Since the n-bit blocks $(M[1], ..., M[q])$ are the same for the two messages, $\sum_{i=1}^{q}(Y[i])$ are also the same for the two messages where each $Y[i]$ means the output of the i-th E cipher (see Fig. 1). We then have the following equation for a collision.

$$E_K((X^*||padding) \oplus \sum_{i=1}^{q}(Y[i]) \oplus 0^n) = E_K(Z^* \oplus \sum_{i=1}^{q}(Y[i]) \oplus L \cdot x^{-1}). \quad (1)$$

Since $E_K(\cdot)$ is a permutation on the set of all n-bit blocks, we get

$$L \cdot x^{-1} = (X^*||padding) \oplus Z^*. \quad (2)$$

Thus, the attacker can compute the key information L from Eq. (2) with no knowledge of K and can use it to forge the MAC of a new message. The attacker chooses a message $M(1)^k = (M[1], ..., M[q], X^k)$ in the first set of messages $M(1)^i$ which does not cause a collision and gets the corresponding MAC value as follows.

$$E_K((X^k||padding) \oplus \sum_{i=1}^{q}(Y[i]) \oplus 0^n). \quad (3)$$

The attacker then also knows the corresponding MAC value for a message $(M[1], ..., M[q], U)$ is the same as the MAC value (Eq. (3)) of the $M(1)^k$, where $U = (X^k||padding) \oplus L \cdot x^{-1}$ and U is n-bit. That is, one can forge the MAC of $(M[1], ..., M[q], (X^k||padding) \oplus L \cdot x^{-1})$ which does not belong to the second set of messages by our assumption. The complexity of this attack is thus $[0, 0, 2^{n/2+1}, 0]$. Note that the tuple $[a, b, c, d]$ introduced in [5] is used to quantify the resources needed for an attack where a denotes the number of off-line block cipher encipherments (or decipherments), b denotes the number of known message string/MAC pairs, c denotes the number of chosen message string/MAC pairs, and d denotes the number of on-line MAC verifications (i.e., to submit a message string/MAC pair and receive an answer indicating whether or not the MAC is valid than to obtain the genuine MAC value for a message).

Taking into account one-block message sets $M(1)^i = X^i$ and $M(2)^j = Z^j$ in the above two message sets, i.e., $q = 0$, we can convert this chosen message attack into a known message attack with a $[0, 2^{n/2+1}, 0, 0]$ complexity. It follows that we can construct a forgery attack with $2^{n/2+1}$ message queries such that the lower bound on the advantage of the forgery attacker for PMAC is $L_b = \frac{(\sigma+1)^2}{2^{n+3}}$, since $\sigma = 2^{n/2+1}$ implies $L_b \approx 0.5 (< 0.63)$ where $\sigma = \sum_{i=1}^{q} \frac{\lceil |M_i| \rceil}{n}$ and M_i are message queries. Note that if we perform this attack with a random function instead of PMAC, the probability of producing a forgery is 2^{-n}, so this probability can be ignored in the computation of the advantage of the forgery attacker. Since $L_b = \frac{1}{16} \cdot U_b$ for the upper bound $U_b = \frac{(\sigma+1)^2}{2^{n-1}}$ in [2], our attack and [2] show that there exists a forgery attacker $\mathcal{A}$ for PMAC such that $\frac{1}{16} \cdot U_b \leq Adv(\mathcal{A}) \leq U_b$, where $Adv(\mathcal{A})$ is the advantage of the $\mathcal{A}$. This does not contradict the security bound shown by Black and Rogaway [2], but shows the tightness of their security bound.

PMAC where truncation is performed ($\tau < n$): Now we consider the case where truncation is used in the PMAC algorithm. As like the foregoing assumptions we suppose that the attacker obtains two sets which are composed of $2^{n/2}$ message/MAC pairs each, denoted $M(1)^i = (M[1], ..., M[q], X^i)$ and $M(2)^j = (M[1], ..., M[q], Z^j)$ where the last blocks X^i and Z^j have the same conditions as those of the above attack.

Similarly, we can expect to find about $2^{n-\tau}$ external collisions (i.e., messages between two message sets with matching MAC values). Let these messages which cause external collisions denote respectively $M(1)^{*(l)} = (M[1], ..., M[q], X^*(l))$ and $M(2)^{*(l)}$=$(M[1], ..., M[q], Z^*(l))$ for $1 \leq l \leq 2^{n-\tau}$. The problem remains to find the internal collision from amongst the many external collisions. If $M(1)^{*(k)} = (M[1], ..., M[q], X^*(k))$ and $M(2)^{*(k)} = (M[1], ..., M[q], Z^*(k))$ for some k $(1 \leq k \leq 2^{n-\tau})$ are the internal collision, as like the previous subsection, we get the following equation.

$$E_K((X^*(k)||padding) \oplus \sum_{i=1}^{q} Y[i] \oplus 0^n) = E_K(Z^*(k) \oplus \sum_{i=1}^{q} Y[i] \oplus L \cdot x^{-1}).$$

So we get

$$L \cdot x^{-1} = (X^*(k)||padding) \oplus Z^*(k). \tag{4}$$

In order to find the index k, i.e., the internal collision, the attacker can perform the following process: First, the attacker computes the candidates of $L \cdot x^{-1}$, denoted $L \cdot x^{-1}(l)$, where $L \cdot x^{-1}(l) = (X^*(l)||padding) \oplus Z^*(l)$, $1 \leq l \leq 2^{n-\tau}$ (from the above $M(1)^{*(l)}$ and $M(2)^{*(l)}$). Second, the attacker chooses a message $(M[1], ..., M[q], X^S)$ in the first set of messages $M(1)^i$. Third, the attacker requires the MAC verifications for the $2^{n-\tau}$ messages $(M[1], ..., M[q], U(l))$ with the MAC value of $(M[1], ..., M[q], X^S)$ where $U(l) = (X^S||padding) \oplus L \cdot x^{-1}(l)$. That is, the attacker checks whether or not the MAC values for $(M[1], ..., M[q], U(l))$ are the same as the MAC value of $(M[1], ..., M[q], X^S)$ by the MAC verifications. If a message $(M[1], ..., M[q], U(l))$ passes the above test, $L \cdot x^{-1}(l)$ is a candidate of $L \cdot x^{-1}$. By this process, the attacker can reduce the $2^{n-\tau}$ candidates of internal collisions to $2^{n-2\cdot\tau}$. If the attacker repeats this process by $\lceil n/\tau \rceil$ times, then the attacker can find the desired internal collision and from this collision the attacker can also obtain $L \cdot x^{-1}$ with no knowledge of K. Thus, the attacker can forge the MAC of a new message using the obtained value of $L \cdot x^{-1}$ as in the previous attack. This attack requires about $2^{n/2+1}$ chosen messages and $2^{n-\tau} + 2^{n-2\cdot\tau} + \cdots + 2^{n-\lceil n/\tau \rceil \cdot \tau} \approx 2^{n-\tau}$ MAC verifications. That is, the complexity of this attack is $[0, 0, 2^{n/2+1}, 2^{n-\tau}]$. Similarly, this attack can be converted into a known message attack with a $[0, 2^{n/2+1}, 0, 2^{n-\tau}]$ complexity (by considering $q = 0$).

3.2 Key Recovery Attacks on PMAC

In order to devise key recovery attacks on PMAC, we ask for MAC values of $\lceil \frac{k}{\tau} \rceil$ one-block messages M_i whose bit-lengths are all less than n, i.e., with the known message attack we obtain (M_1, T_1), (M_2, T_2), $\cdots$, $(M_{\lceil \frac{k}{\tau} \rceil}, T_{\lceil \frac{k}{\tau} \rceil})$, where each T_i is the first τ bits of $E_K(pad(M_i))$. We use the obtained message/MAC pairs to do an exhaustive search for the key. If τ is larger than k, it requires about 2^{k-1} block cipher E encryptions on average to recover the key. Otherwise, it requires about $2^k + 2^{k-\tau} + \cdots + 2^{k-(\lceil \frac{k}{\tau} \rceil - 1)\tau} \approx 2^k$ block cipher E encryptions to recover the key since each message/MAC pair offers a τ-bit restriction. Hence, if $\tau > k$ then the complexity of the attack is $[2^{k-1}, 1, 0, 0]$, otherwise, $[2^k, \lceil \frac{k}{\tau} \rceil, 0, 0]$.

4 Forgery and Key Recovery Attacks on TMAC-V

In this section we describe forgery and key recovery attacks on a CBC-MAC variant, TMAC-V, with known or chosen message queries.

4.1 Forgery Attacks on TMAC-V

Our TMAC-V attacks also start from collecting enough message/MAC pairs to get a collision. With the known message attack the attacker obtains the corresponding MACs for approximately $2^{n/2}$ different one-block messages $X^i (1 \leq i \leq 2^{n/2})$ where X^i whose sizes are less than n bits are pairwise-distinct. Note that all these messages require padding. The attacker also obtains the corresponding

MACs for approximately $2^{n/2}$ different one-block messages $Z^j (1 \leq j \leq 2^{n/2})$ where Z^j whose sizes are equal to n bits are pairwise distinct.

Using the birthday paradox arguments [17], with a high probability, approximately 0.63, the attacker expects to find at least one external collision between two message sets. Suppose the pair of messages which cause a collision are respectively X^* and Z^* where $|X^*| < n$ and $|Z^*| = n$. Then, by the definition of TMAC-V, the attacker has the following equation.

$$E_K((X^*||padding) \oplus E_{K'}(S_3)) = E_K(Z^* \oplus E_{K'}(S_2)). \tag{5}$$

Since $E_K(\cdot)$ is a permutation,

$$(X^*||padding) \oplus Z^* = E_{K'}(S_2) \oplus E_{K'}(S_3). \tag{6}$$

The attacker can use Eq. (6) to forge the MAC of a new message as follows: The attacker chooses a message X^S in the first set of messages which does not cause a collision. By the definition of TMAC-V, the MAC value of the X^S is $E_K((X^S||padding) \oplus E_{K'}(S_3))$. The attacker then also knows that the corresponding MAC value for a message $((X^S||padding) \oplus ((X^*||padding) \oplus Z^*))$ whose size is n bits is the same as the MAC value of X^S. Thus, the attacker can forge the MAC of a new message $((X^S||padding) \oplus ((X^*||padding) \oplus Z^*))$ using $2^{n/2+1}$ known messages. Therefore, the complexity of this attack is $[0, 2^{n/2+1}, 0, 0]$. It is easy to see that we are able to use multi-block message queries (as in the PMAC attacks) to devise a chosen message attack on TMAC-V with a $[0, 0, 2^{n/2+1}, 0]$ complexity.

4.2 Key Recovery Attacks on TMAC-V

We present here two attacks to recover the whole key of TMAC-V; one uses Eq. (5) and Eq. (6) derived from the above forgery attacks on TMAC-V and the other uses the meet-in-the-middle technique.

As stated above, we use $2^{n/2+1}$ known messages to get a collision, i.e., Eq. (5) and Eq. (6). If $n > k$, it requires about 2^k block cipher E encryptions on average to recover the key K' since Eq. (6) asks two encryptions of E for each candidate of the key K'. Once we recover the key K', we again use Eq. (5) to recover the remaining key K. Since we know the input and output pair of E_K, it requires about 2^{k-1} block cipher E encryptions on average to recover the key K. Hence, the complexity of this attack is $[3 \cdot 2^{k-1}, 2^{n/2+1}, 0, 0]$. If $n < k$, in order to recover the key K' from Eq. (6) we should require $\lceil k/n \rceil$ collisions satisfying Eq. (6). So the data complexity of this attack is $\lceil \sqrt{\lceil k/n \rceil} \rceil \cdot 2^{n/2+1}$ known messages from which we expect more than $\lceil k/n \rceil$ collisions (the cardinality of the first message set is the same as that of the second message set, $\lceil \sqrt{\lceil k/n \rceil} \rceil \cdot 2^{n/2}$). Similarly, we apply these collisions to Eq. (6) and Eq. (5) for finding K' and K, respectively. Since it requires $2 \cdot (2^k + 2^{k-n} + \cdots + 2^{k-(\lceil k/n \rceil - 1) \cdot n}) \approx 2^{k+1}$ E encryptions to recover the key K' and $2^k + 2^{k-n} + \cdots + 2^{k-(\lceil k/n \rceil - 1) \cdot n} \approx 2^k$ E encryptions to recover the key K, the complexity of this attack is $[3 \cdot 2^k, \lceil \sqrt{\lceil k/n \rceil} \rceil \cdot 2^{n/2+1}, 0, 0]$.

The meet-in-the-middle technique also allows to recover the key of TMAC-V. First, we encrypt S_2 through E using all candidates of K' and keep all the encrypted values with respect to key candidates in a table, called $\mathcal{S}$ (this step requires 2^k n-bit blocks and 2^k k-bit blocks of memory together with 2^k E encryptions). Second, we ask for MAC values of $\lceil 2k/n \rceil$ one-block messages whose bit-lengths are all less than n, i.e., with the known message attack we obtain message/MAC pairs (M_1, T_1), (M_2, T_2), $\cdots$, $(M_{\lceil 2k/n \rceil}, T_{\lceil 2k/n \rceil})$. Third, we decrypt T_1 through E using all candidates of K and keep all the decrypted values with respect to key candidates in a table, called $\mathcal{T}$ (this step also requires 2^k n-bit blocks and 2^k k-bit blocks of memory together with 2^k E decryptions) and we update $\mathcal{S}$ by adding M_1 with each of the stored values in $\mathcal{S}$. In this step we discard keys which do not match between $\mathcal{S}$ and $\mathcal{T}$ (this step can efficiently be done by sorting the tables $\mathcal{S}$ and $\mathcal{T}$). We expect about 2^{2k-n} keys to be remained after this step. Fourth, we also decrypt T_2 through E using the candidates of K and update $\mathcal{T}$ with all the decrypted values with respect to key candidates K, and we again update $\mathcal{S}$ by adding $M_1 \oplus M_2$ with each of the stored values in $\mathcal{S}$. We then expect 2^{2k-2n} keys to be remained after this step. We repeatedly do this step for all the remaining message/MAC pairs one by one, then after all steps we recover K and K' since the expectation of the number of remaining keys is $2^{2k-\lceil 2k/n \rceil \cdot n} < 1$ (the right key is not discarded). The total time complexity is less than $(\lceil 2k/n \rceil + 1) \cdot 2^k$ E encryptions and thus the complexity of this attack is approximately $[(\lceil 2k/n \rceil + 1) \cdot 2^k, \lceil 2k/n \rceil, 0, 0]$ with 2^{k+2} storage.

5 Conclusion

PMAC is a fully parallelizable alternative to the CBC-MAC, and Michell's TMAC variant, TMAC-V, is an improvement of TMAC. In this paper we have studied the security of PMAC and TMAC-V against the forgery and the key recovery attacks. Tables 1 and 2 summarize our attacks together with previous attacks on other MAC algorithms. From the tables, we clearly know that, in terms of security, PMAC (which is similar in functionality to the OMAC) and TMAC-V do not offer significant advantages in comparison with XCBC, TMAC, OMAC and EMAC. In particular, TMAC-V does not have a good advantage over TMAC against the forgery and the full key recovery attacks although it removes a simple algebraic relationship between two keys to improve TMAC, and our attacks on PMAC do not contradict the security proof but establish the tightness of the security bound.

Acknowledgments

We would like to thank the anonymous referees and C. Mitchell for helpful comments about this work. 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 a Ph.D.

Table 1. Forgery Attack Complexities

Scheme	[a, b, c, d]	Condition	Paper
PMAC	$[0\ ,2^{n/2+1},0,\ 0]$	no truncation	This paper
PMAC	$[0\ ,\ 2^{n/2+1},0,\ 2^{n-\tau}]$	τ-bit trucation	This paper
TMAC-V	$[0\ ,\ 2^{n/2+1},\ 0,\ 0]$	no truncation	This paper
XCBC	$[0\ ,\ 2^{n/2+1},\ 0,\ 0]$	no truncation	[15, 16]
TMAC	$[0\ ,\ 2^{n/2+1},\ 0,\ 0]$	no truncation	[15, 16]
OMAC	$[0\ ,\ 2^{n/2+1},\ 0,\ 0]$	no truncation	[15, 16]
EMAC	$[0\ ,\ 2^{n/2},\ 1,\ 0]$	no truncation	[5]

Table 2. Key Recovery Attack Complexities

Scheme	$[a,\ b,\ c,\ d]$	Condition	Paper
PMAC	$[2^{k-1}, 1, 0, 0]$	$\tau > n$	This paper
PMAC	$[2^k, \lceil \frac{k}{\tau} \rceil, 0, 0]$	$\tau \leq n$	This paper
TMAC-V	$[3 \cdot 2^{k-1}, 2^{n/2+1}, 0, 0]$	P-O attack($n > k$)	This paper
TMAC-V	$[3 \cdot 2^k, \lceil \sqrt{\lceil \frac{k}{n} \rceil} \rceil \cdot 2^{n/2+1}, 0, 0]$	P-O attack($n \leq k$)	This paper
TMAC-V	$[(\lceil \frac{2k}{n} \rceil + 1) \cdot 2^k, \lceil \frac{2k}{n} \rceil, 0, 0]$	MIMA(2^{k+2} storage)	This paper
XCBC	$[2^k,\ 2^{n/2+1},\ 0,\ 0]$	P-O attack	[15, 16]
XCBC	$[2^{k+1},\ \lceil \frac{k+2n}{n} \rceil,\ 0,\ 0]$	MIMA(negligible storage)	[15, 16]
TMAC	$[2^k,\ 2^{n/2+1},\ 0,\ 0]$	P-O attack	[15, 16]
TMAC	$[2^{k+1},\ \lceil \frac{k+n}{n} \rceil,\ 0,\ 0]$	MIMA(negligible storage)	[15, 16]
OMAC	$[2^k\ ,\ 2^{n/2+1},\ 0,\ 0]$	P-O attack	[15, 16]
OMAC	$[2^{k+1}\ ,\ \lceil \frac{k}{n} \rceil,\ 0,\ 0]$	MIMA(negligible storage)	[15, 16]
EMAC	$[2^{k+1}\ ,\ 2^{n/2},\ 0,\ 0]$	P-O attack	[5]
EMAC	$[s \cdot 2^k\ ,\ \lceil \frac{2k}{n} \rceil,\ 0,\ 0]$	MIMA($O(2^k)$ storage)	[5]

[1] MIMA: the meet-in-middle attack
[2] P-O attack: the attack based on the Preneel-van Oorschot attack [20]

grant of the Katholieke Universiteit Leuven and by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD) (KRF-2005-213-D00077) 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.

References

1. K. Brincat and C. Mitchell, *New CBC-MAC Forgery Attack*, ACISP 2001, LNCS 2119, pp. 3-14, Springer-Verlag, 2001.
2. J. Black and P. Rogaway, *A Block-Cipher Mode of Operation for Parallelizable Message Authentication*, Advances in Cryptology – EUROCRYPT 2002 , LNCS 2332, pp. 384-397, Springer-Verlag, 2002.
3. M. Bellare, J. Kilian and P. Rogaway, *The Security of the Cipher Block Chaining Message Authentication Code*, Advances in Cryptology – CRYPTO 1994, LNCS 839, pp. 341-358, Springer-Verlag, 1994.

4. J. Black and P. Rogaway, *CBC-MACs for Arbitrary-Length Messages : The Three Key Construction*, Advances in Cryptology – CRYPTO 2000, LNCS 1880, pp. 197-215, Springer-Verlag, 2000.
5. ISO/IEC 9797-1, *Information technology – Security techniques – Message Authentication Codes (MACs)– Part 1 : Mechanisms using a block cipher*, International Organization for Standardization, Geneve, Swizerland, 1999.
6. T. Iwata and K. Kurosawa, *OMAC: One-Key CBC MAC*, FSE 2003, LNCS 2887, pp. 137-162, Springer-Verlag, 2003.
7. T. Iwata and K. Kurosawa, *On the Security of Two New Omac Variants*, ICISC 2003, LNCS 2971, pp. 67-78, Springer-Verlag, 2004.
8. T. Iwata and K. Kurosawa, *Strong security bounds for OMAC, TMAC, XCBC*, Indocrypt 2003, LNCS 2904, pp. 402-415, Springer-Verlag, 2003.
9. E. Jaulmes, A. Joux and F. Valette, *On the Security of Randomized CBC-MAC beyond the Birthday Limit: A New Construction*, FSE 2002, LNCS 2365, pp. 237-251, Springer-Verlag, 2002.
10. T. Kohno, *Related-Key and Key-Collision Attacks against RMAC*, Cryptology ePrint Archive, 2002. Available at http://eprint.iacr.org. Also available at http://csrc.nist.gov/CryptoToolkit/modes/comments.
11. K. Kurosawa and T. Iwata, *TMAC: Two-Key CBC-MAC*, Topics in Cryptology – CT-RSA 2003, LNCS 2612, pp. 33-49, Springer-Verlag, 2003.
12. L. Knudsen and C. Mitchell, *Analysis of 3GPP-MAC and two-key 3GPP-MAC*, Discrete Appiled Mathematics , pp. 181-191, Elsevier Science, 2003.
13. L. Knudsen and C. Mitchell, *Partial key recovery attack against RMAC*, Journal of Cryptology, Vol. 18, No. 4, pp. 375-389, 2005.
14. L. Knudsen and T. Khono, *Analysis of RMAC*, FSE 2003, LNCS 2887, pp. 191-200, Springer-Verlag, 2003.
15. C. Mitchell, *On the security of XCBC, TMAC, and OMAC*, Technical Report RHCL-MA-2003-4, Aug. 2003. Available at http://www.rhul.ac.uk/mathematics /techreports or http://csrc.nist.gov/CryptoToolkit/modes/comments.
16. C. Mitchell, *Partial Key Recovery Attacks on XCBC, TMAC and OMAC*, Cryptography and Coding, 10th IMA International Conference – CCC 2005, LNCS 3796, pp. 155-167, Springer-Verlag, 2005.
17. A. Menezes, P.C. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boer Raton, 1997.
18. National Institute of Standards and Technology (NIST), Gaithersburg, MD. *NIST Special Publication 800-38B, Draft Recommendation for Block Cipher Modes of Operation: the RMAC Authentication*, November, 2002.
19. E. Petrank and C. Rackoff, *CBC MAC real-time data sources*, Journal of Cryptology, Vol. 13, No. 3, pp. 315-348, Springer-Verlag, 2000.
20. B. Preneel and P.C. van Oorschot, *On the security of iterated Message Authentication Codes*, IEEE Transactions on Information Theory, Vol. 45, No. 1, pp. 188-199, 1999.
21. J. Sung, D. Hong and S. Lee, *Key Recovery Attacks on the RMAC, TMAC, and IACBC*, ACISP 2003, LNCS 2727, pp. 265-273, Springer-Verlag, 2003.

Side Channel Attacks Against HMACs Based on Block-Cipher Based Hash Functions*

Katsuyuki Okeya

Hitachi, Ltd., Systems Development Laboratory,
1099, Ohzenji, Asao-ku, Kawasaki, 215-0013, Japan
ka-okeya@sdl.hitachi.co.jp

Abstract. HMAC is one of the most famous keyed hash functions, and widely utilized. In order to design secure hash functions, we often use PGV construction consisting of 64 schemes, each of which utilizes a block cipher. If the underlying block cipher is ideal, 12 schemes are proven to be secure. In this paper, we evaluate the security of these schemes in view of side channel attacks. As it turns out, HMACs based on 11 out of 12 secure PGV schemes are vulnerable to side channel attacks, *even if the underlying block cipher is secure against side channel attacks.* These schemes are classified into two groups based on their vulnerabilities. For the first group which contains 8 schemes, we show that the attacker can reveal the whole key of HMAC, and selectively forge in consequence. For the other group which contains 3 schemes, we specify the importance of the execution sequence for the inner operations of the scheme, and refine it. If wrong orders of operations are used, the attacker can reveal a portion of the key of HMAC. Hence, the use of HMACs based on such PGV schemes as they are is not recommended when the resistance against side channel attacks is necessary.

Keywords: (keyed) hash function, HMAC, PGV construction, side channel attacks, differential power analysis (DPA), reverse DPA.

1 Introduction

Nowadays, on mobile devices, digital communications such as electronic commerce are not special any more, or rather they are a part of our daily life. In order to securely communicate between such mobile devices, the counterpart is often authenticated. Thus, cryptographic algorithms are implemented on the device. However, since the computational resources for cryptosystems on the device are normally scarce or limited, a lightweight implementation is preferable, on the one hand. Under these situations, a keyed hash function such as HMAC [BCK96] is often utilized for authentication.

* This work was partly supported by National Institute of Information and Communications Technology (NICT).

L. Batten and R. Safavi-Naini (Eds.): ACISP 2006, LNCS 4058, pp. 432–443, 2006.
© Springer-Verlag Berlin Heidelberg 2006

On the other hand, whenever cryptographic algorithms are implemented on the device, the resistance against side channel attacks should be taken into account, not only the mathematical security of the algorithms. Hence, it is important to research the side channel attacks against HMAC.

Lemke et al. [LSP04] discussed a side channel attack against HMAC with RIPEMD-160 [DBP96], and it may be applicable to HMAC with SHA-1 [SHA]. However, this approach was ad-hoc and tailored to specific hash functions, and its extent is not clear. In view of the construction of a secure hash function against side channel attacks, the design principle of the hash functions should be taken into account. Hence, in more general settings, the discussion on side channel attacks against HMACs is needed, apart from specific hash functions.

In this paper, we evaluate the resistance of HMACs based on block-cipher based hash functions against side channel attacks, when PGV construction [PGV94] for designing the hash functions is utilized. According to PGV construction, twelve schemes are proven to have collision resistance and onewayness. We will show that eleven out of twelve are vulnerable to side channel attacks, especially differential power analysis (DPA), *even if the underlying block cipher is secure against side channel attacks*. These schemes are classified into two groups based on their vulnerabilities. For the first group which contains eight schemes, we will show that the attacker can reveal the whole key of HMAC, and selectively forge in consequence. For the other group which contains three schemes, we specify the importance of the execution sequence for the inner operations of the compression function, and refine it. In addition, we will propose a DPA attack named *reverse DPA*. If wrong orders of operations are used, the attacker can reveal a portion of the key of HMAC under the reverse DPA. Hence, the use of HMACs based on such PGV schemes as they are is not recommended when the resistance against side channel attacks is necessary.

The rest of this paper is organized as follows. Section 2 recalls hash functions and PGV construction. Section 3 recalls side channel attacks and DPA, and describes DPA model. Section 4 applies DPA attacks against HMACs based on PGV construction. Section 5 concludes this paper.

2 Hash Functions

For a given data with arbitrary length, hash functions[1] computes a fixed length data, namely hashed value. As the security of hash functions, *onewayness* and *collision resistance* are required. It is said that a hash function *hash* has *onewayness* if for any value y in the set of all the outputs, it is hard to find out an input x with $hash(x) = y$. It is said that a hash function *hash* has *collision resistance* if it is hard to find out a pair of values x, y in the set of all the inputs such that the relation $hash(x) = hash(y)$ is hold.

[1] Whereas "hash function" stands for a *oneway* hash function in some cryptographic contexts, the onewayness is not included in the definition of a hash function in this paper for the simplicity of the terminologies.

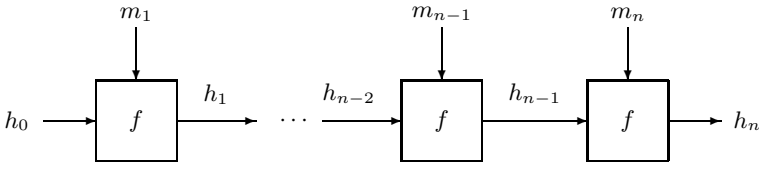


Fig. 1. An iterated hash function

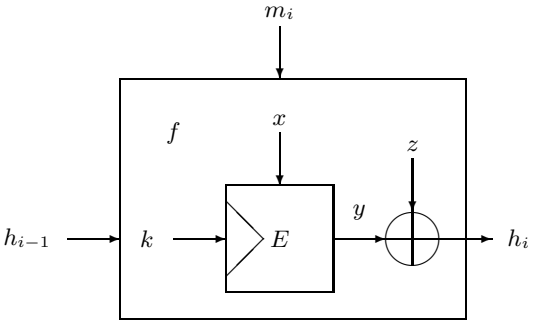


Fig. 2. A compression function based on a block cipher

In order to construct secure hash functions, compression functions are often used. A compression function converts a given data with a fixed length to a data with a shorter length. Hash functions repeatedly utilize compression functions for converting a given arbitrary-length data to a fixed-length data. An iterated hash function utilizes a compression function sequentially for computing the hashed value. Figure 1 shows an iterated hash function, where f stands for a compression function, $m_1, \cdots, m_{n-1}, m_n$ for fixed length portions of the input message m, h_0 for initial value, $h_1, \cdots, h_{n-1}$ for intermediate values, h_n for the output of the hash function $hash(m)$. In the case of the iterated hash function, if the underlying compression function has collision resistance, the hash function also has collision resistance [Dam89, Mer89].

In several approaches to design compression functions, we often utilize a construction method based on block ciphers. The combinations with input (h_{i-1}, m_i) and output $h_i (= f(h_{i-1}, m_i))$ of the compression function f, and input (x, key k) and output $y (= E_k(x))$ of the underlying block cipher E provide us with various compression functions.[2]

Preneel-Govaerts-Vandewalle (PGV) construction [PGV94] utilizes $x, k, z \in \{h_{i-1}, m_i, h_{i-1} \oplus m_i, 0\}$. Thus, there are 64 schemes in total. In these schemes, 12 schemes have collision resistance and onewayness if the underlying block cipher is ideal [BRS02]. These schemes are listed at Figure 8 in appendix.

[2] In this paper, we assume the size of m_i is equal to that of h_i. However, in general, this is not the case. Actually, in many hash functions such as SHA-1 [SHA], these sizes are different, and they do not use XOR operations for these values such as $m_i \oplus y$.

Some applications of hash functions are keyed hash functions such as HMAC [BCK96]. In a keyed hash function, a part of its input is secret.

3 Side Channel Attacks

Side channel attacks (SCA) are a serious menace to embedded devices with cryptographic applications. Because while such devices perform cryptographic operations, execution timing, power consumption and suchlike, namely side channel information, are observable for an attacker, and he/she can utilize the observed information for detecting the secret stored in the device. Examples of side channel attacks are timing attack [Koc96], simple power analysis (SPA) [KJJ99], differential power analysis (DPA) [KJJ99]. In this paper, we focus on power analysis, especially DPA, for the evaluation of keyed hash functions. The same applies to timing attack or electro-magnetic analysis. In that case, different kind of side channel information is only utilized, and the analysis method is not so different. Since the way of observing the side channel information is independent from implemented cryptographic algorithms, it is not necessary to discuss it with focusing on hash functions.

In a differential power analysis (DPA), the attacker observes power consumption as side channel information, and tries to reveal the secret using statistical tools such as the average of power consumption for eliminating the noise [KJJ99]. Because of the use of statistical tools, the attacker observes power consumption several time.

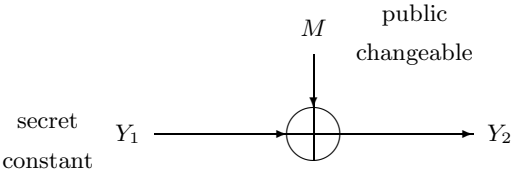


Fig. 3. DPA model

DPA Model. Since hash functions as they are do not have any secret information, the application of side channel attacks against hash functions requires us to combine hash functions and secret, namely keyed hash function. In fact, even if the same underlying hash function is utilized, the existence and the role of the secret for the hash function depend on the way to apply to the upper schemes. Thus, we consider how to apply side channel attacks on a target operation in the hash function.

We explain the way to apply DPA on a target operation using the following setting [OI05]. In this setting, while we assume that the target operation is XOR, the same applies to other operations such as arithmetic addition.

Figure 3 shows a target XOR operation in a keyed hash function. The XOR operation has two inputs Y_1 and M; Y_1 is secret and constant which the attacker tries to reveal, and M is public and the attacker can control this value. Y_2 is the output of the XOR, and the attacker does not know this value.

The first step of the attack is to guess a certain bit b of Y_1, and he/she sorts the input M depending on the target bit of Y_2 is 0 or 1 when the changeable input M is inputted; $\mathcal{M}_b = \{M|(\text{the } i\text{-th bit of } M) \oplus b = 0\}$ and $\mathcal{M}_{\bar{b}} = \{M|(\text{the } i\text{-th bit of } M) \oplus b = 1\}$ for b, the guessed i-th bit of Y_1.

Then, the attacker observes the power consumption for the XOR operation with several[3] inputs M; power consumption P_M for $M \in \mathcal{M}^u$, the set of utilized M for observations.

The third step is to compute the average power consumption for each group; $AP_b = \frac{1}{\#\mathcal{M}_b^u} \sum_{M \in \mathcal{M}_b^u} P_M$ and $AP_{\bar{b}} = \frac{1}{\#\mathcal{M}_{\bar{b}}^u} \sum_{M \in \mathcal{M}_{\bar{b}}^u} P_M$, where $\mathcal{M}_b^u = \mathcal{M}_b \cap \mathcal{M}^u$ and $\mathcal{M}_{\bar{b}}^u = \mathcal{M}_{\bar{b}} \cap \mathcal{M}^u$. Under the Hamming weight model [MDS99], the power consumption depends on the Hamming weight of manipulated data. Hence a large power consumption, that is $AP_{\bar{b}}(t) \gg AP_b(t)$ for time t when the XOR operation is performed, implies that the target bit of Y_2 is 1 since the other bits behave as random and the averaging eliminates their effect. This provides the attacker with the information whether the original guess b for the target bit of the secret Y_1 is correct or not.

Repeating this procedure, the attacker can reveal the whole bits of the secret Y_1. Note that once the attacker observes sufficient number of the power consumptions, he/she does not have to re-observe them for another target bit. The only thing he/she has to do is to re-classify M and compute the average power consumption for the new groups.

4 Side Channel Attacks on Compression Functions

In this section, we consider side channel attacks on HMACs based on block-cipher based hash functions. Section 4.1 discusses target compression functions of HMAC. Section 4.2 applies the DPA attack on twelve secure PGV schemes. We assume that the underlying block cipher is resistant against side channel attacks. Because the resistance against side channel attacks on the block ciphers can be discussed independently from the hash functions. Besides, lots of researches have been done [MDS99, Mes00a, Mes00b]. Section 4.3 proposes the reverse DPA, which is another model of DPA. Section 4.4 discusses forgery and key recovery of HMACs using the DPA attacks.

4.1 Target Compression Functions in HMAC

First, we discuss the target compression functions of the DPA attack in the case of the message authentication code HMAC, one of the keyed hash functions.

In HMAC, for a given message m, using the key K, the message authentication code is computed as follows:

$$\text{HMAC}(m) = hash(K \oplus opad || hash(K \oplus ipad || m)),$$

[3] The required number of samples is determined by the signal-to-noise (S/N) ratio of the power consumption on the target device.

where *ipad* and *opad* are constant values. We denote by IV the initial vector of the hash function *hash*. Using the compression function f, two values K_{in} and K_{out} are defined as follows:

$$K_{in} = f(IV, K \oplus ipad), \quad K_{out} = f(IV, K \oplus opad).$$

Roughly speaking, HMAC computes the message authentication code using two related keys K_{in} and K_{out}.

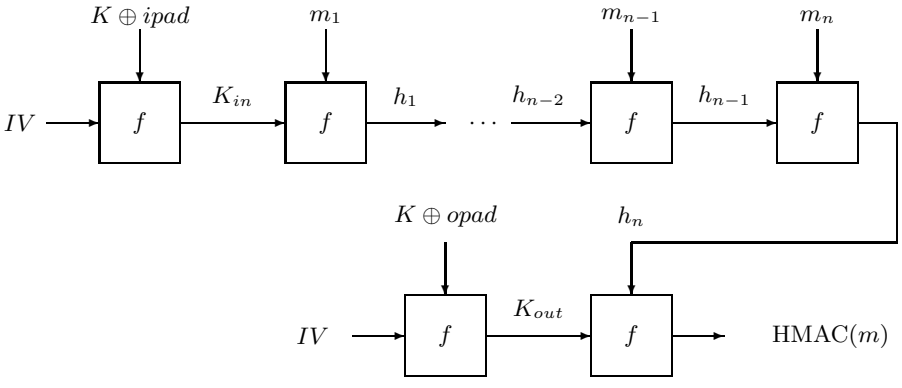


Fig. 4. HMAC

Figure 4 shows the algorithm of HMAC. The message m is divided into the fixed-length blocks $m_1, \cdots, m_n$. Then HMAC performs the compression function f $n+3$ times. $m_1, \cdots, m_n$ are inputted as a part of message input of the first application of the hash function *hash*. The output $h_n (= hash(K \oplus ipad || m))$ of the first application of the hash function *hash* is inputted as a part of message input of the second application of the hash function *hash*. The output $hash(K \oplus opad || h_n)$ of the second application of the hash function *hash* is the output HMAC(m) of HMAC for the message m.

In HMAC, K_{in} and K_{out} are secret and constant values. The message m is public for the attacker, and we may assume he/she can freely change this value. When $m_1, \cdots, m_{i-1}$ are fixed, the output h_{i-1} of the i-th compression function $f(h_{i-2}, m_{i-1})$ in the first application of the hash function *hash* is constant. Note that the value h_{i-1} is secret for the attacker, and he/she can forge a message authentication code by using this value if the compression function is insecure. Besides, h_{i-1} is equal to the key K_{in} if $i = 1$. On the other hand, m_i is public for the attacker, and he/she can change this value independent from $m_1, \cdots, m_{i-1}$.

Thus, we assume the following setting: In the compression function $f(h_{i-1}, m_i)$, h_{i-1} is secret and fixed, and m_i is public and changeable. The goal of the attacker is to reveal h_{i-1} using DPA attack with the help of the power consumption relating the output h_i.

4.2 DPA Against PGV Compression Functions

Based on PGV construction, there are 64 schemes, that is, all the combinations of input/output of a compression function and input/output of the underlying block cipher of the compression function. There are 12 out of 64 schemes which have collision resistance and onewayness, and the list is at Figure 8 in appendix.

We discuss side channel attacks against these twelve compression functions. Note that, it is assumed that the underlying block cipher is secure against side channel attacks. Thus, side channel attacks on the block cipher are not considered. The DPA attack discussed in Section 3 is applied to the above twelve schemes.

We consider the following situation: The attacker feeds $m_1, m_2, \cdots, m_{i-1}$ as fixed inputs and m_i as a changeable input to the target hash function. In this case, m_i is public and changeable and h_{i-1} is secret and fixed for him/her, as discussed in the previous section.

If there exists an XOR operation $m_i \oplus h_{i-1}$ in the compression function, the attacker applies the DPA attack to this operation, and he/she can reveal the secret h_{i-1}.

In the above twelve schemes, eight schemes have the XOR operation $m_i \oplus h_{i-1}$; $f_2, f_4, f_6, f_8, f_9, f_{10}, f_{11}, f_{12}$. If these schemes are utilized for designing a hash function, the hash function is vulnerable to the DPA attack. In order to evaluate the remaining four schemes f_1, f_3, f_5, f_7, we will enhance the DPA attack in the next section.

4.3 Reverse DPA

In Section 3, we discussed the way to mount the DPA attack against the XOR operation. In this section, we will enhance it on the following setting.

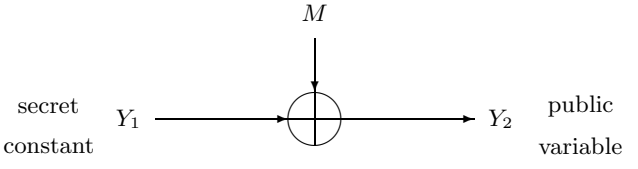


Fig. 5. Reverse DPA

Figure 5 shows a target XOR operation[4] in a keyed hash function. The XOR operation has two inputs Y_1 and M; Y_1 is secret and constant which the attacker tries to reveal, and M is also secret, but he/she does not care. Y_2 is the output of the XOR, and is public and variable, but it is not assumed that he/she can control.

[4] In this paper, we mainly discuss a side channel attack on the XOR operation. But it is applicable to some other operations such as the arithmetic addition, which is utilized in many hash functions such as SHA-1 [SHA] and MD5 [MD5]. In the case of the arithmetic addition, the target bit is given from the least significant bit in turn.

In what follows, we propose a DPA attack on the above XOR, and it is referred to as *reverse DPA*. In the reverse DPA, *time goes back*. That is, for detecting the secret Y_1, instead of using the input M, the attacker uses the output Y_2 and guesses the backward value M and observes the power consumption.

An example of such an XOR operation is that in the case of HMAC with the compression function f_5, the XOR operation of the final compression function in it. (See Figures 4 and 8.) The output Y_2 corresponds to the output of HMAC, which is public and variable. The input Y_1 corresponds to the secret K_{out}, which is constant as long as the initial vector IV and the key K are constant. The other input M corresponds to h_n, which is secret.

The first step of the reverse DPA is to guess a certain bit of Y_1, and he/she sorts the output Y_2 depending on the target bit of the input M is 0 or 1 according to the variable output Y_2. Then, the attacker observes the power consumption for the XOR operation with several outputs Y_2. The third step is to compute the average power consumption for each group, and confirms the correctness of the original guess using the averages.

In fact, the attacker cannot perform in the above order, since at the time he/she gets the output Y_2, the XOR operation was over. In reality, first, he/she observes the power consumption for some input M, and gets the output Y_2. Based on the obtained Y_2, he/she selects one of two groups for the output Y_2. He/she repeats this, and gets enough samples. Then, he/she computes the average power consumption for each group.

We experimented the reverse DPA on some IC chip. A toy program that performs the XOR operation was implemented on it, and the power consumption was observed $10,000$ times. Figure 6 shows the differences between the average power consumptions. The upper half is the case that the target bit is 1, and the lower half is the case that it is 0. We can confirm the clear spikes. The leftmost spikes are the target one, and these *signs* are different. It means that the attacker can retrieve the target bit of the secret. The next three spikes correspond to the save of registers, since some registers store the manipulated data that are related with the secret.[5] Hence, the reverse DPA is realistic.

4.4 DPA Against HMAC

This section shows forgery and key recovery of HMACs using the DPA attacks discussed in Sections 4.2 and 4.3 against the compression functions.

The case f_j with $j \in \{2, 4, 6, 8, 9, 10, 11, 12\}$. We consider forgery of HMAC with the compression function f_j. The attacker selects m_1 as a public changeable value. Since m_1 is public and changeable and K_{in} is secret and constant for him/her, the DPA attack in Section 4.2 is applicable to the compression function $f_j(K_{in}, m_1)$. Then he/she reveals the key K_{in}.

The next target for him/her is to reveal the key K_{out}. For the revealed key K_{in}, the attacker computes the output h_n of the first application of the hash

[5] The upper half has a thick spike in the rightmost side. We could not find out the reason why it appears. It may be a ghost spike.

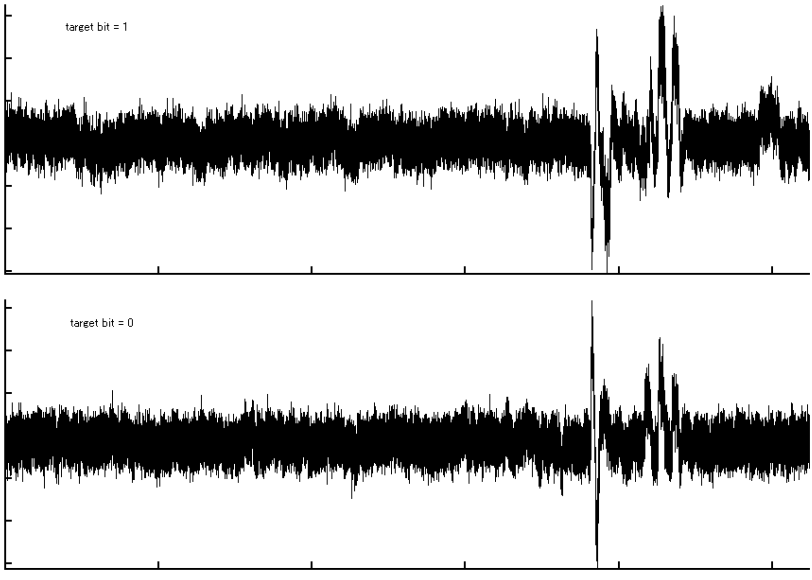


Fig. 6. Experiment of the reverse DPA with 10,000 samples

function for various messages m. While the output h_n is public for the attacker, it is hard for him/her to select a message m with a pre-determined h_n, if the hash function has onewayness. On the other hand, in the DPA model, inputs are only classified depending on the target bit is 0 or 1. Thus, in the average power consumption, the effects from the other bits are cancelled if the input behaves as a random number. Hence, if we regard the output of the hash function as a random number, "the input M is changeable" in the DPA model is not a problem when he/she computes sufficiently many h_n. Therefore, the DPA model is applicable to the compression function $f_j(K_{out}, h_n)$ in the second application of the hash function of HMAC, and the attacker can reveal the key K_{out}.

Once he/she reveals the keys K_{in} and K_{out}, he/she can compute HMAC(m) for any message m. In other words, the attacker can *selectively forge* if these eight PGV schemes are utilized for HMAC.

The case f_j with $j \in \{3, 5, 7\}$. We consider the reverse DPA against the remaining four PGV schemes under the HMAC setting; the final compression function in the second application of the hash function, that is, the output h_i(=HMAC(m)) is public and variable, one of the inputs $h_{i-1}(= K_{out})$ is secret and constant, and the other input $m_i(= h_n)$ is secret.

First, in the case of the compression function f_1, there is no target XOR operation for the reverse DPA. Thus, f_1 is secure. Next, in the case of the compression function f_5, the reverse DPA is applicable to the XOR operation. Thus, the attacker can reveal the secret K_{out}. In the case of the compression

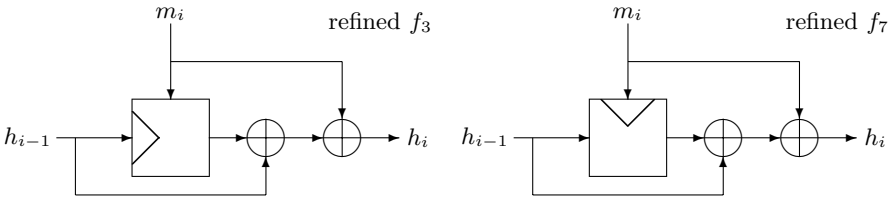


Fig. 7. Refined compression functions

functions f_3 and f_7, they have the XOR operation with three inputs, and the order of the XOR operations is important;

$$(1)\ h_i = (m_i \oplus h_{i-1}) \oplus y \quad (2)\ h_i = (y \oplus m_i) \oplus h_{i-1} \quad (3)\ h_i = (y \oplus h_{i-1}) \oplus m_i.$$

In the case of (1), the DPA attack in Section 4.2 is applicable to the XOR operation $m_i \oplus h_{i-1}$. Thus it is vulnerable to the DPA. In the case of (2), the reverse DPA attack is applicable to the XOR operation $M \oplus h_{i-1} = h_i$, where M stands for $y \oplus m_i$. Thus, the attacker can reveal h_{i-1}, that is the key K_{out}. In the case of (3), neither normal nor reverse DPA attack is applicable. Thus, it still survives under these attacks. Hence, we have refined the compression functions f_3 and f_7 as specified at Figure 7.

As it turns out, except for f_1, refined f_3 and refined f_7, the attacker can *recover the key* K_{out} of the HMAC with f_j with $j \in \{3, 5, 7\}$.

Remark 1. The compression function f_5 is used for SHA-1/2 [SHA] and MD5 [MD5], and RIPEMD-160 [DBP96] uses a duplicated f_5.

Remark 2. Whether or not forgery of HMAC is possible using the revealed key K_{out} is an open problem.

5 Conclusion

In view of side channel attacks, we have evaluated the security of twelve PGV schemes that have collision resistance and onewayness if the underlying block cipher is ideal. As it turned out, HMACs based on eleven out of twelve shemes are vulnerable to side channel attacks, *even if the underlying block cipher is secure against side channel attacks*. For the compression functions f_j with $j \in \{2, 4, 6, 8, 9, 10, 11, 12\}$, we have shown that the attacker can reveal the keys K_{in} and K_{out} of HMAC, and selectively forge in consequence. For the compression functions f_j with $j \in \{3, 5, 7\}$, we have specified the importance of the execution sequence for the inner XOR operations of the compression function, and refined them as refined f_3 and f_7. If wrong orders of XOR operations are used, the attacker can reveal the key K_{out} of HMAC under the reverse DPA. Therefore, the use of HMACs based on such schemes as they are, even if they have collision resistance and onewayness, is not recommended when the resistance against side channel attacks is necessary.

Acknowledgment

We would like to thank Mr. Hirotaka Yoshida, Dr. Tetsu Iwata, and Mr. Camille Vuillaume for valuable comments on the previous version of this paper.

References

[BCK96] Bellare, M., Canetti, R., Krawczyk, H., *Keying Hash Functions for Message Authentication*, Advances in Cryptology - CRYPTO '96, LNCS1109, (1996), 1-15.

[BRS02] J. Black, P. Rogaway, and T. Shrimpton, *Black-box analysis of the block cipher-based hash-function constructions from PGV*, Advances in Cryptology - CRYPTO 2002, Springer-Verlag, LNCS 2442, (2002), 320-335.

[Dam89] Damgård, I., *A design priciple for hash functions*, Advances in Cryptology - CRYPTO '89, LNCS435, (1990), 416-427.

[DBP96] Dobbertin, H., Bosselaers, A., Preneel, B., *RIPEMD-160: A Strengthened Version of RIPEMD*, Fast Software Encryption (FSE '96), LNCS 1039, (1996), 71-82.

[Koc96] Kocher, C., *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Advances in Cryptology - CRYPTO '96, LNCS1109, (1996), 104-113.

[KJJ99] Kocher, C., Jaffe, J., Jun, B., *Differential Power Analysis*, Advances in Cryptology - CRYPTO '99, LNCS1666, (1999), 388-397.

[LSP04] Lemke, K., Schramm, K., Paar, C., *DPA on n-Bit Sized Boolean and Arithmetic Operations and Its Application to IDEA, RC6, and the HMAC-Construction*, Cryptographic Hardware and Embedded Systems (CHES 2004), LNCS3156, (2004), 205-219.

[MD5] RFC 1321, *The MD5 Message-Digest Algorithm*, (1992).

[MDS99] Messerges, T.S., Dabbish, E.A., Sloan, R.H., *Investigations of Power Analysis Attacks on Smartcards*, USENIX Workshop on Smartcard Technology, (1999).

[Mer89] Merkle, R., *One way hash functions and DES*, Advances in Cryptology - CRYPTO '89, LNCS435, (1990), 428-446.

[Mes00a] Messerges, T.S., *Securing the AES Finalists Against Power Analysis Attacks*, Fast Software Encryption (FSE 2000), LNCS1978, (2000), 150-164.

[Mes00b] Messerges, T.S., *Using Second-Order Power Analysis to Attack DPA Resistant Software*, Cryptographic Hardware and Embedded System (CHES 2000), LNCS1965, (2000), 238-251.

[OI05] K. Okeya, T. Iwata, *Side Channel Attacks against Message Authentication Codes*, 2nd European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS 2005), LNCS3813, (2005), 205-217.

[PGV94] B. Preneel, R. Govaerts, and J. Vandewalle, *Hash functions based on block ciphers: A synthetic approach*, Advanced in Cryptology, CRYPTO '93, Springer-Verlag, LNCS 773, (1994), 368-378.

[SHA] FIPS PUB 180-2, *Secure Hash Standard (SHS)*, (2002).

A Compression Functions Based on PGV Construction

We list twelve schemes base on PGV construction which have collision resistance and onewayness if the underlying block cipher is ideal.

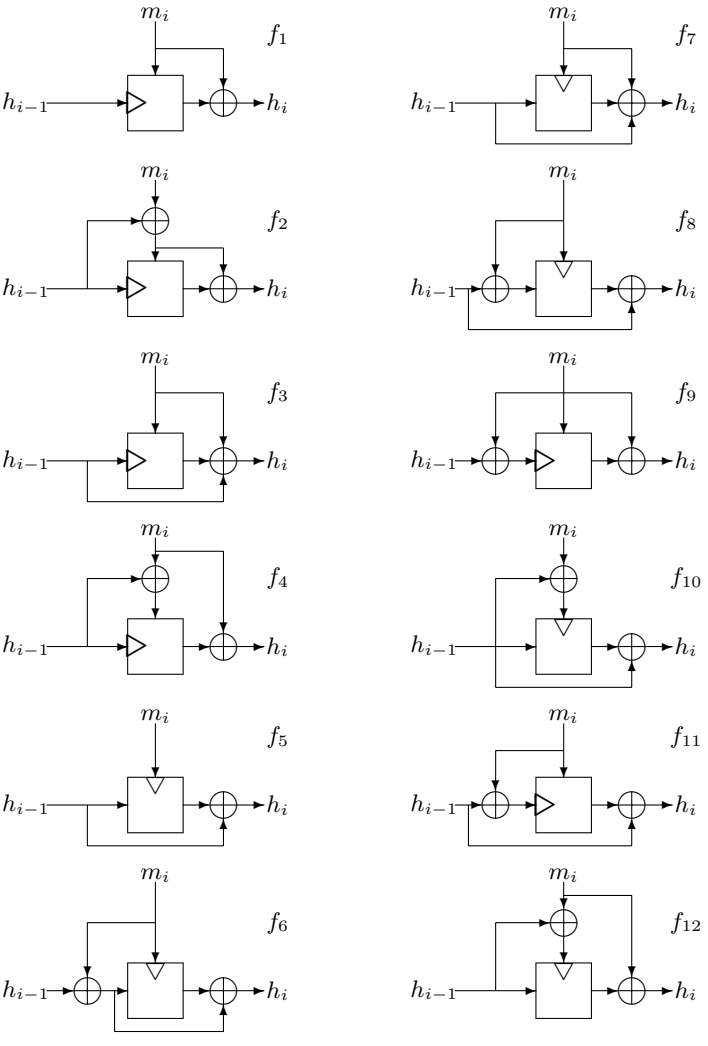


Fig. 8. Compression functions based on PGV construction

Author Index

Abe, Masayuki 360
Adelsbach, André 136
Agudo, Isaac 383
Al-Hinai, Sultan 1
Au, Man Ho 223

Bao, Feng 148, 313
Bao, Li 395
Batten, Lynn 1
Boyd, Colin 247, 371
Braeken, An 40
Burmester, Mike 295

Caelli, William J. 183
Cao, Zhenfu 195
Cho, Joo Yeon 29
Choo, Kim-Kwang Raymond 371
Colbert, Bernard 1
Courtois, Nicolas T. 76
Cui, Yang 360

Dawson, Ed 52, 247, 407
Debraize, Blandine 76
Deng, Xiaotie 235
Duan, Shanshan 195

Fayyoumi, Ebaa 324
Fournier, Jacques 17

Galindo, David 336
Garrido, Eric 76
Gauravaram, Praveen 407

Hanaoka, Goichiro 123, 348
Hayashi, Ryotaro 271
Henricksen, Matt 52, 64
Holford, John W. 183
Hong, Seokhie 421
Hu, Bessie C. 235
Huber, Ulrich 136

Imai, Hideki 123, 348, 360

Jiang, Shaoquan 259

Kiltz, Eike 336
Kim, Jongsung 421
Kitagawa, Takashi 348
Kurosawa, Kaoru 360

Lakshminarayanan, A. 87
Lano, Joseph 40
Le, Tri Van 295
Lee, Changhoon 421
Lee, Sangjin 421
Leiwo, Jussipekka 171
Lim, T.L. 87
Lin, Yue-Hsun 159
Lopez, Javier 383

Mathuria, Anish 371
Matsuura, Kanta 348
Medeiros, Breno de 295
Millan, William 407
Montenegro, Jose A. 383
Mu, Yi 99

Ogawa, Kazuto 123
Okamoto, Eiji 247
Okeya, Katsuyuki 432
Oommen, B. John 324
Overbeck, R. 283

Paterson, Kenneth G. 207
Peng, Kun 247
Pieprzyk, Josef 29
Premkumar, Benjamin 171
Preneel, Bart 40

Quynh, Nguyen Anh 111

Sadeghi, Ahmad-Reza 136
Schuldt, Jacob C.N. 207
Simpson, Leonie 64
Sun, Hung-Min 159
Sung, Jaechul 421
Susilo, Willy 99, 223

Takefuji, Yoshiyasu 111
Tanaka, Keisuke 271
Tunstall, Michael 17

Viswanathan, Kapali 407

Watanabe, Hajime 348
Wong, Duncan S. 235
Wong, Kenneth 1
Wu, Ming-Fung 159

Xu, Shidi 99

Yang, Peng 348
Yin, Yin 395
Yiu, Siu-Ming 223
Yu, Yu 171

Zhang, Rui 348
Zhang, Zhenfeng 235
Zhu, Huafei 148, 313