Ronald Cramer (Ed.)

LNCS 3494

Advances in Cryptology – EUROCRYPT 2005

24th Annual International Conference on the Theory and Applications of Cryptographic Techniques
Aarhus, Denmark, May 2005, Proceedings

Springer

Lecture Notes in Computer Science 3494

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

Editorial Board

Ronald Cramer (Ed.)

Advances in Cryptology – EUROCRYPT 2005

24th Annual International Conference on the Theory and Applications of Cryptographic Techniques
Aarhus, Denmark, May 22-26, 2005
Proceedings

Volume Editor

Ronald Cramer
CWI, Amsterdam
and Mathematical Institute, Leiden University
Kruislaan 413, P.O. Box 94079
1090GB Amsterdam, The Netherlands
E-mail: cramer@cwi.nl, cramer@math.leidenuniv.nl

Library of Congress Control Number: 2005926095

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

ISSN 0302-9743
ISBN-10 3-540-25910-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-25910-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11426639 06/3142 5 4 3 2 1 0

Table of Contents

Cryptanalysis I

Theory I

Encryption I

Signatures and Authentication

Algebra and Number Theory I

Quantum Cryptography

Secure Protocols

Algebra and Number Theory II

Theory II

Encryption II

Cryptanalysis II

Broadcast Encryption and Traitor Tracing

Preface

These are the proceedings of the 24th Annual IACR Eurocrypt Conference. The conference was sponsored by the International Association for Cryptologic Research (IACR; see `www.iacr.org`), this year in cooperation with the Computer Science Department of the University of Aarhus, Denmark. As General Chair, Ivan Damgård was responsible for local organization.

The Eurocrypt 2005 Program Committee (PC) consisted of 30 internationally renowned experts. Their names and affiliations are listed on pages VII and VIII of these proceedings. By the November 15, 2004 submission deadline the PC had received a total of 190 submissions via the IACR Electronic Submission Server. The subsequent selection process was divided into two phases, as usual. In the review phase each submission was carefully scrutinized by at least three independent reviewers, and the review reports, often extensive, were committed to the IACR Web Review System. These were taken as the starting point for the PC-wide Web-based discussion phase. During this phase, additional reports were provided as needed, and the PC eventually had some 700 reports at its disposal. In addition, the discussions generated more than 850 messages, all posted in the system. During the entire PC phase, which started in August 2003 with my earliest invitations to PC members and which continued until March 2005, more than 1000 email messages were communicated. Moreover, the PC received much appreciated assistance from a large body of external reviewers. Their names are listed on page VIII of these proceedings.

The selection process for Eurocrypt 2005 was finalized by the end of January 2005 with a one-day PC meeting held in Amsterdam, The Netherlands. This meeting was attended by most of the PC members. The PC ultimately selected 33 papers for publication in these proceedings and presentation at the conference. After notification of acceptance the authors were provided with the review comments and were granted one month to prepare the final versions, which were due by February 28, 2005. These final versions were not subjected to further scrutiny by the PC and their authors bear full responsibility.

It was a great pleasure to work with this PC, and I thank all members for contributing so much of their scientific expertise, advice, opinions, preferences and devotion, and for their very hard work in the relatively short time frame that a PC has to operate in.

The Eurocrypt 2005 "Best Paper Award" was shared by Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen and Xiuyuan Yu for their paper "Cryptanalysis of the Hash Functions MD4 and RIPEMD" and by Xiaoyun Wang and Hongbo Yu for their paper "How to Break MD5 and Other Hash Functions."

Besides the above-mentioned 33 regular presentations, the Eurocrypt 2005 scientific program featured two invited speakers: *René Schoof* (University of Rome, Italy), with a survey talk on algebraic geometry algorithms in cryptology,

in particular on point counting algorithms for algebraic varieties over finite fields, and *Joe Kilian* (Yianilos Labs, Princeton, USA), with a talk on "Confusion, Quagmire and Irrelevancy: an Optimist's View of the Future of Cryptographic Research."

Many others have, in one way or another, helped the PC, contributed to these proceedings or the Eurocrypt conference as such, thereby also serving the international cryptology community as a whole, directly or indirectly.

The Eurocrypt conference continues to attract many very high-quality submissions from all over the world; so many in fact that not all good papers could be selected. All authors who submitted their work for consideration by the PC are hereby acknowledged for their contributions.

CWI[1] in Amsterdam and the Mathematical Institute at Leiden University, my employers, are gratefully acknowledged for their support.

Eurocrypt 2004 PC Co-chairs Christian Cachin and Jan Camenisch (IBM Research), as well as Crypto 2004 PC Chair Matt Franklin (UC Davis), gave useful advice on a number of occasions. Also many thanks to Springer for its collaboration. Peter Landrock (Cryptomathic) is kindly acknowledged for agreeing to organize and chair the Eurocrypt 2005 rump session, a traditional, entertaining Tuesday evening session with brief research announcements and "any other business."

Hats off to John Tromp (Quantum Computing and Advanced Systems Research Group, CWI), who reallocated, from the summer of 2004 until February 2005, substantial amounts of his precious research time to expertly manage the technical infrastructure for electronic submissions and Web review. The software was run on the network of CWI's INS Department. I hereby acknowledge the support of INS head Martin Kersten and his system manager Matthijs Mourits. Also many thanks to Harry Buhrman and Paul Vitányi! Thomas Herlea from KU Leuven's IACR submission server and webreview system development team offered prompt technical assistance to John whenever needed. Michael Smeding (Computer Support Team, CWI) provided prompt service to me and my group.

Serge Fehr of my Cryptology and Information Security Research Group at CWI was in charge of "General Affairs." In particular, he assisted me during the very busy week following the submission deadline, organized the PC meeting in collaboration with Wilmy van Ojik (Conference Organization, CWI), helped the PC by logging the entire decision process during the meeting, and provided instrumental assistance when I edited this volume. Serge, thanks a lot!

Finally, I thank Ivan Damgård, Eurocrypt 2005 General Chair, for our very pleasant collaboration during the organization of Eurocrypt 2005, a memorable addition to our many joint scientific endeavors (and friendship!)

March 2005 Ronald Cramer

[1] CWI is the National Research Institute for Mathematics and Computer Science in The Netherlands.

EUROCRYPT 2005

May 22–26, 2005, Aarhus, Denmark

Sponsored by the
International Association for Cryptologic Research (IACR)
in cooperation with the
Computer Science Department, Faculty of Science,
University of Aarhus, Denmark

General Chair

Ivan Damgård, Department of Computer Science,
University of Aarhus, Denmark

Program Chair

Ronald Cramer, CWI, Amsterdam & Mathematical Institute,
Leiden University, The Netherlands

Program Committee

Michael Backes ..... IBM Zürich Research Laboratory, Switzerland
Daniel Bleichenbacher ..... Lucent Bell Labs, USA
Don Beaver ..... Syntechnica, LLC, USA
Don Coppersmith ..... IBM T. J. Watson Research Center, USA
Hans Dobbertin ..... University of Bochum, Germany
Yevgeniy Dodis ..... New York University, USA
Marc Fischlin ..... ETH Zürich, Switzerland
Steven Galbraith ..... Royal Holloway, University of London, UK
Shafi Goldwasser ..... MIT, USA & Weizmann Institute of Science, Israel
Shai Halevi ..... IBM T. J. Watson Research Center, USA
Johan Håstad ..... Royal Institute of Technology, UK
Marc Joye ..... Gemplus, France
Aggelos Kiayias ..... University of Connecticut, USA
Eyal Kushilevitz ..... Technion, Israel
Arjen Lenstra ..... Lucent Bell Labs, USA & TU Eindhoven, The Netherlands
Phong Q. Nguyễn ..... CNRS & École Normale Supérieure, France
Kaisa Nyberg ..... Nokia, Finland
Tatsuaki Okamoto ..... NTT, Japan
Rafail Ostrovsky ..... U.C.L.A., USA
Carles Padró ..... Universitat Politècnica de Catalunya, Spain
Benny Pinkas ..... Hewlett-Packard Labs, Israel

..... *(continued on the next page)* .....

Bart Preneel ..... Katholieke Universiteit Leuven, Belgium
Louis Salvail ..... University of Aarhus, Denmark
Palash Sarkar ..... Indian Statistical Institute, India
Berry Schoenmakers ..... TU Eindhoven, The Netherlands
Igor Shparlinski ..... Macquarie University, Australia
Douglas Stinson ..... University of Waterloo, Canada
Salil Vadhan ..... Harvard University, USA
Moti Yung ..... Columbia University, USA

External Referees

Michel Abdalla
Masayuki Abe
Saurabh Aggarwal
Roberto Avanzi
Joonsang Baek
Paulo Barreto
Amos Beimel
Eli Biham
Alex Biryukov
Alexandra Boldyreva
Emmanuel Bresson
Éric Brier
Christian Cachin
Jan Camenisch
Ran Canetti
Christophe De Cannière
Dario Catalano
Debrup Chakraborty
Yan-Cheng Chang
Denis Charles
Sanjit Chatterjee
Benoît Chevallier-Mames
Olivier Chevassut
Scott Contini
Giovanni Di Crescenzo
Ivan Damgård
Drew Dean
Jean-François Dhem
Iwan Duursma
Stefan Dziembowski
Kirsten Eisentraeger
Nelly Fazio
Matthias Fitzi
Pierre-Alain Fouque
Matt Franklin
Michael H. Freedman
Atsushi Fujioka
David Galindo
Juan Garay
Rosario Gennaro
Guang Gong
Maribel González Vasco
Ignacio Gracia
Louis Granboulan
Stuart Haber
Helena Handschuh
Alex Healy
Javier Herranz
Florian Hess
Jason Hinek
Martin Hirt
Susan Hohenberger
Thomas Holenstein
Nick Howgrave-Graham
Yuval Ishai
Markus Jakobsson
Stanislaw Jarecki
Antoine Joux
Ari Juels
Jonathan Katz
Alexander Kholosha
Eike Kiltz
Tetsutaro Kobayashi
Tadayoshi Kohno
Yuichi Komano
Hugo Krawczyk
Gunnar Kreitz
Caroline Kudla
Noboru Kunihiro
Jeff Lagarias
Tanja Lange
Joseph Lano
Kristin Lauter
Yehuda Lindell
Helger Lipmaa
Moses Liskov
Phil MacKenzie
Subhamoy Maitra
Tal Malkin
John Malone-Lee
Stefan Mangard
Keith Martin
Alexander May
Mira Meyerovich
Silvio Micali
Anton Mityagin
Paz Morillo
Siguna Mueller
Sourav Mukhopadhyay
Enric Nart
Kenny Nguyen
Minh-Huyen Nguyen
Antonio Nicolosi
Jesper Nielsen
Kobbi Nissim
Satoshi Obana
Miyako Ohkubo
Kazuo Ohta
Elisabeth Oswald
Pascal Paillier
Rafael Pass
Kenny Paterson

Maura Paterson
Souradyuti Paul
Thomas Pedersen
Jan Pelzl
Giuseppe Persiano
Erez Petrank
Birgit Pfitzmann
Duong Hieu Phan
Krzysztof Pietrzak
David Pointcheval
Manoj Prabhakaran
Bartosz Przydatek
Jordi Pujolàs
Tal Rabin
Omer Reingold
Rennato Renner
Leonid Reyzin
Vincent Rijmen
Pankaj Rohatgi
Alon Rosen
Germán Sáez
Kazue Sako
Takakazu Satoh
Christian Schaffner
Werner Schindler
Mike Scott
Hovav Shacham
Ronen Shaltiel
Peter Shor
Victor Shoup
Tom Shrimpton
Alice Silverberg
Nigel Smart
Martijn Stam
François-Xavier Standaert
Allan Steel
Damien Stehlé
Ron Steinfeld
Koutarou Suzuki
Mike Szydlo
Keisuke Tanaka
Tamir Tassa
Yael Tauman
Isamu Teranishi
Edlyn Teske
Mårten Trolin
Yiannis Tsiounis
Pim Tuyls
Shigenori Uchiyama
Vinod Vaikuntanathan
Ingrid Verbauwhede
Frederik Vercauteren
Eric Verheul
Jorge Luis Villar
Michael Waidner
Shabsi Walfish
Huaxiong Wang
Xiaoyun Wang
Mark Watkins
Benne de Weger
Steve Weis
Annegret Weng
Mike Wiener
Douglas Wikström
Christopher Wolf
Stefan Wolf
Go Yamamoto
Aleksandr Yampolskiy
Yuliang Zheng
Hong-Sheng Zhou

Cryptanalysis of the Hash Functions MD4 and RIPEMD

Xiaoyun Wang[1], Xuejia Lai[2], Dengguo Feng[3], Hui Chen[1], and Xiuyuan Yu[4]

[1] Shandong University, Jinan250100, China
xywang@sdu.edu.cn
[2] Shanghai Jiaotong University, Shanghai200052, China
[3] Chinese Academy of Science China, Beijing100080, China
[4] Huangzhou Teacher College, Hangzhou310012, China

Abstract. MD4 is a hash function developed by Rivest in 1990. It serves as the basis for most of the dedicated hash functions such as MD5, SHAx, RIPEMD, and HAVAL. In 1996, Dobbertin showed how to find collisions of MD4 with complexity equivalent to 2^{20} MD4 hash computations. In this paper, we present a new attack on MD4 which can find a collision with probability 2^{-2} to 2^{-6}, and the complexity of finding a collision doesn't exceed 2^8 MD4 hash operations. Built upon the collision search attack, we present a chosen-message pre-image attack on MD4 with complexity below 2^8. Furthermore, we show that for a weak message, we can find another message that produces the same hash value. The complexity is only a single MD4 computation, and a random message is a weak message with probability 2^{-122}.

The attack on MD4 can be directly applied to RIPEMD which has two parallel copies of MD4, and the complexity of finding a collision is about 2^{18} RIPEMD hash operations.

1 Introduction

MD4 [14] is an early-appeared hash function that is designed using basic arithmetic and Boolean operations that are readily available on modern computers. Such type of hash functions are often referred to as dedicated hash functions, and they are quite different from hash functions based on block ciphers. After the publication of MD4, several dedicated hash functions are successively designed, including MD5 [15], HAVAL [18], RIPEMD [13], RIPEMD-160 [9], SHA-1 [10], SHA-256 [11], etc. These hash functions, although more complex, all follow the same design philosophy as MD4 and have similar structures as MD4. In particular, RIPEMD consists of two parallel copies of MD4, and each copy is identical to MD4 except for some internal constants.

There have been several important cryptanalytical results for both MD4 and RIPEMD. In 1996, H. Dobbertin [5] gave a collision attack on MD4 which finds a collision with probability 2^{-22}. He also showed how to find collisions of meaningful messages. In 1998, H. Dobbertin [8] showed that the first two (out of the

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 1–18, 2005.

total three) rounds of MD4 is not one-way, and this means that there is an efficient attack for finding a preimage and a second preimage. For RIPEMD, H. Dobbertin [7] gave an attack that finds a collision of RIPEMD reduced to two rounds with 2^{31} hash operations.

Along with the development of the MD4-family of hash functions, there have also been security analysis on these functions. For example, B. den Boer and A. Bosselaers [3] found pseudo-collisions (same message with two different initial values) for MD5. In Eurocrypto'96, H. Dobbertin [6] presented a collisions of MD5, under another initial value. In Crypto'98, F. Chabaud and A. Joux [4] presented a differential attack on SHA-0 with probability 2^{-61}. At Asiacrypt 2003, B.V. Rompay etc. [16] gave a collision attack on HAVAL-128 with probability 2^{-29}.

Some very interesting results on hash functions came out simultaneously in Crypto 2004. Eli Biham and Rafi Chen [2] presented a near-collision attack on SHA-0, and described their improved results on SHA-0 and SHA-1 in the rump session. Then, A. Joux [12] presented a real collision of SHA-0 with four message blocks. X.Y. Wang etc. [17] also announced real collisions of a series of hash functions including MD4, MD5, HAVAL-128, and RIPEMD in the rump session. All these research work were done independently.

The purpose of this paper is to analyze the security of MD4 and RIPEMD and present more efficient attacks. The main results are summarized below.

1. Collision search attack on MD4: we can find collisions with probability 2^{-2} to 2^{-6} and with complexity less than 2^8 MD4 hash operations.
2. A theoretical second pre-image attack on MD4 for weak messages: For a weak message, we can find another message that produces the same hash value. The complexity is only a single MD4 computation and a random selected message is a weak message with probability 2^{-122}.
3. Collision search attack on RIPEMD: we can find collisions with probability 2^{-16} and with complexity less than 2^{18} RIPEMD hash operations.

In addition to presenting the new attacks on MD4 and RIPEMD, we also introduce a set of new analytical techniques that are applicable to all the hash functions in the MD4-family. More specifically, we show how to derive a set of the sufficient conditions on the chaining values to ensure the differential path to hold, and how to use message modification techniques to greatly improve the success probability of the attack. Such techniques have proved to be very effective in cryptanalyzing other dedicated hash functions such as MD5, RIPEMD, HAVAL-128, HAVAL-160, SHA0, and especially SHA-1.

All the existing attacks on dedicated hash functions belong to differential attacks [1], since a collision can be regarded as a special differential which has non-zero input difference and zero output difference. We remark that unlike other existing attacks on hash functions, our attack presented in this paper is a "precise" differential attack in which the differential path is more restrictive since it depends on both the difference as well as the specific value of the bit involved.

The paper is organized as follows. In Section 2 we provide a description of MD4 and RIPEMD. In Section 3, we summarize some useful properties of the Boolean functions in two hash functions and introduce the notation used in the paper. As our main result, the collision attack on MD4 is presented in Section 4, the collision attack on RIPEMD is presented in Section 5. In Section 6, we describe a theoretical second pre-image attack on MD4. In Section 7, we summarize our work together with some remarks, especially on the implication for the analysis of the hash function SHA-0.

2 Description of MD4 and RIPEMD

2.1 MD4 Algorithm

The message digest algorithm MD4 compresses any arbitrary bit-length message into a 128-bit hash value. Given any message, the algorithm first pads it into a message with a length that is a multiple of 512 bits We omit the padding method here since it is irrelevant to our attack.

For each 512-bit message block, MD4 compresses it into a 128-bit hash value using a compression function. The MD4 compression function has three rounds. Each round uses a different nonlinear Boolean function defined as follows:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$

Here X, Y, Z are 32-bit words. The operations of three functions are all bitwise. $\neg X$ is the bitwise complement of X, $\wedge$, $\oplus$ and $\vee$ are respectively the bitwise AND, XOR and OR.

Each round of the compression function repeats 16 similar step operations, and in each step one the four chaining variables a, b, c, d is updated.

$$\phi_0(a,b,c,d,m_k,s) = ((a + F(b,c,d) + m_k) \bmod 2^{32}) \lll s$$
$$\phi_1(a,b,c,d,m_k,s) = ((a + G(b,c,d) + m_k + \texttt{0x5a827999}) \bmod 2^{32}) \lll s$$
$$\phi_2(a,b,c,d,m_k,s) = ((a + H(b,c,d) + m_k + \texttt{0x6ed9eba1}) \bmod 2^{32}) \lll s$$

The initial value for MD4 is defined as:

$$(a,b,c,d) = (\texttt{0x67452301},\ \texttt{0xefcdab89},\ \texttt{0x98badcfe},\ \texttt{0x10325476})$$

MD4 Compression Function. For one 512-bit block M of the padded message $\overline{M}$, $M = (m_0, m_1, ..., m_{15})$, the compression function is defined as follows:

1. Let (aa, bb, cc, dd) be input chaining variables for M. If M is the first message block to be hashed, then (aa, bb, cc, dd) are set to be the initial value. Otherwise they are the output from compressing the previous message block.

2. Perform the following 48 steps (three rounds):
   For $j = 0, 1, 2$ and $i = 0, 1, 2, 3$

$$\begin{aligned} a &= \phi_j(a, b, c, d, w_{j,4i}, s_{j,4i}) \\ d &= \phi_j(d, a, b, c, w_{j,4i+1}, s_{j,4i+1}) \\ c &= \phi_j(c, d, a, b, w_{j,4i+2}, s_{j,4i+2}) \\ b &= \phi_j(b, c, d, a, w_{j,4i+3}, s_{j,4i+3}) \end{aligned}$$

   Here $s_{j,4i+k}$ $(k = 0, 1, 2, 3)$ are step-dependent constants, $w_{j,4i+k}$ is a message word and $\lll s_{j,4i+k}$ is circularly left-shift by $s_{j,4i+k}$ bit positions. The specific message order and shift positions are given in Table 5.
3. Add the chaining variables a, b, c and d respectively to the input chaining variables to produce the final chaining variables for the current message block.

$$\begin{aligned} aa &= (a + aa) \bmod 2^{32} \\ bb &= (b + bb) \bmod 2^{32} \\ cc &= (c + cc) \bmod 2^{32} \\ dd &= (d + dd) \bmod 2^{32} \end{aligned}$$

If M is the last message block, $H(\overline{M}) = aa|bb|cc|dd$ is the hash value for the message $\overline{M}$. Otherwise repeat the above process with the next 512-bit message block and (aa, bb, cc, dd) as the input chaining variables.

2.2 RIPEMD Algorithm

RIPEMD employs the same nonlinear round functions as MD4 and they are used in the following six operations:

$$\begin{aligned} \varphi_0(a, b, c, d, m_k, s) &= ((a + F(b, c, d) + m_k) \bmod 2^{32}) \lll s \\ \varphi_1(a, b, c, d, m_k, s) &= ((a + G(b, c, d) + m_k + \texttt{0x5a827999}) \bmod 2^{32}) \lll s \\ \varphi_2(a, b, c, d, m_k, s) &= ((a + H(b, c, d) + m_k + \texttt{0x6ed9eba1}) \bmod 2^{32}) \lll s \\ \psi_0(a, b, c, d, m_k, s) &= ((a + F(b, c, d) + m_k + \texttt{0x50a28be6}) \bmod 2^{32}) \lll s \\ \psi_1(a, b, c, d, m_k, s) &= ((a + G(b, c, d) + m_k) \bmod 2^{32}) \lll s \\ \psi_2(a, b, c, d, m_k, s) &= ((a + H(b, c, d) + m_k + \texttt{0x5c4dd124}) \bmod 2^{32}) \lll s \end{aligned}$$

In order to easily describe the RIPEMD compression function, we denote MD4 compression function with three operations ϕ_0, ϕ_1 and ϕ_2 as MD4$(\phi_0, \phi_1, \phi_2, M)$.

RIPEMD Compression Function. The RIPEMD compression function employs two copies of MD4 compression function: the left copy is MD4 $(\varphi_0, \varphi_1, \varphi_2, M)$, and the right copy is MD4$(\psi_0, \psi_1, \psi_2, M)$. Both copies have the same initial value as MD4. The details of the message order and shift positions are given in Table 7.

1. Let (a, b, c, d) be the input chaining variables for M which is the same as MD4.

2. Perform two copies of the MD4 operation

$$(aa, dd, cc, bb,) \leftarrow \text{MD4}(\varphi_0, \varphi_1, \varphi_2, M),$$
$$(aaa, ddd, ccc, bbb) \leftarrow \text{MD4}(\psi_0, \psi_1, \psi_2, M).$$

3. The output (a, b, c, d) for compressing M is the following:

$$a = (b + cc + ddd) \bmod 2^{32}$$
$$b = (c + dd + aaa) \bmod 2^{32}$$
$$c = (d + aa + bbb) \bmod 2^{32}$$
$$d = (a + bb + ccc) \bmod 2^{32}$$

3 Preliminaries

3.1 Basic Properties of the Boolean Functions

Some properties of three nonlinear Boolean functions are very helpful for determining sufficient conditions for the differential paths that are used in our collision search attack on MD4 and RIPEMD. In what follows, we summarize some well-known properties of these functions.

Proposition 1. *For the nonlinear function $F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$ in the first round, there are the following properties:*

1. $F(x, y, z) = F(\neg x, y, z)$ *if and only if* $y = z$.
2. $F(x, y, z) = F(x, \neg y, z)$ *if and only if* $x = 0$.
3. $F(x, y, z) = F(x, y, \neg z)$ *if and only if* $x = 1$.

Proposition 2. *For the nonlinear function $G(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$ in the second round, there are the following properties:*

1. $G(x, y, z) = G(\neg x, y, z)$ *if and only if* $y = z$.
2. $G(x, y, z) = G(x, \neg y, z)$ *if and only if* $x = z$.
3. $G(x, y, z) = G(x, y, \neg z)$ *if and only if* $x = y$.

Proposition 3. *For the nonlinear function $H(X, Y, Z) = X \oplus Y \oplus Z$ in the third round, there are the following properties:*

1. $H(x, y, z) = \neg H(\neg x, y, z) = \neg H(x, \neg y, z) = \neg H(x, \neg y, z)$
2. $H(x, y, z) = H(\neg x, \neg y, z) = H(x, \neg y, \neg z) = H(\neg x, y, \neg z)$

3.2 Notation

Here we introduce the notation used in our analysis. Since our attack is a "precise" differential attack, we need to keep track of both the difference as well as the specific value of the bit involved. Therefore, the notation may seem quite complex at a first glance, but the intuition behind these notation will become more clear as we proceed in describing the attacks.

1. $M = (m_0, m_1, ..., m_{15})$ and $M' = (m'_0, m'_1, ..., m'_{15})$ represent two 512-bit messages.
2. a_i, d_i, c_i, b_i respectively denote the outputs of the $(4i-3)$-th, $(4i-2)$-th, $(4i-1)$-th and $4i$-th steps for compressing M, for $1 \leq i \leq 16$.
3. a'_i, b'_i, c'_i, d'_i respectively denote the outputs of the $(4i-3)$-th, $(4i-2)$-th, $(4i-1)$-th and $4i$-th steps for compressing M'.
4. $\Delta m_i = m'_i - m_i$ denotes the difference between two message words m_i and m'_i.
5. $a_{i,j}$, $b_{i,j}$, $c_{i,j}$, $d_{i,j}$ represent respectively the $j-th$ bit of a_i, b_i, c_i, d_i, where the least significant bit is the 1-st bit, and the most significant bit is 32-th bit.
6. $x_i[j]$, $x_i[-j]$ (x can be a, b, c, d) is the resulting values by only changing the $j-th$ bit of the word x_i. $x_i[j]$ is obtained by changing the j-th bit of x_i from 0 to 1. $x_i[-j]$ is obtained by changing the j-th bit of x_i from 1 to 0.
7. $x_i[\pm j_1, \pm j_2, ..., \pm j_l]$ is the value by change $j_1 - th$, $j_2 - th$, ..., $j_l - th$ bits of x_i. The "+" sign means that the bit is changed from 0 to 1, and the "–" sign means that the bit is changed from 1 to 0.

Note that we use integer modular subtraction difference as the measure of difference, not the exclusive-or difference. In addition, we also need to specify the precise values of *each bit* when considering the carry effect in the differential path. This is better understood using an example. Let us consider step 7 in Table 5. The output difference is

$$\Delta c_2 = c'_2 - c_2 = -2^{18} + 2^{21}.$$

Using our notation, $c'_2 = c_2[-19, 22]$. For the specific differential path, we need to expand the *one-bit* subtraction difference in bit 19 into a *three-bit* difference in bits 19,20,21. That is, we expand $c_2[19]$ as $c_2[19, 20, -21]$. Hence, the output c'_2 is represented as

$$c'_2 = c_2[19, 20, -21, 22],$$

as showed in the last column of Table 5.

4 The Collision Attack on MD4

In this section, we will describe a collision attack on MD4 with a success probability 2^{-2} to 2^{-6}. The complexity is below 2^8 MD4 computations. The attack includes three parts:

1. Find a collision differential in which M and M' produces a collision.
2. Derive a set of sufficient conditions which ensure the collision differential to hold.
3. For any random message M, make some modification to M such that almost all the sufficient conditions hold.

4.1 The Collision Differential for MD4

We select a collision differential for MD4 as follows:

$$\Delta H_0 = 0 \overset{(M,M')}{\longrightarrow} \Delta H = 0$$

such that

$$\triangle M = M' - M = (\triangle m_0, \triangle m_1, ..., \triangle m_{15})$$
$$\triangle m_1 = 2^{31},\ \triangle m_2 = 2^{31} - 2^{28},\ \triangle m_{12} = -2^{16}$$
$$\triangle m_i = 0,\ 0 \le i \le 15,\ i \ne 1, 2, 12.$$

All the characteristics in the collision differential can be found in Table 5. The first column denotes the step, the second column is the chaining variable in each step for M, the third is the message word for M in each step, the fourth is shift rotation, the fifth and the sixth are respectively the message word difference and chaining variable difference for M and M', and the seventh is the chaining variable for M'. Especially, the empty items both in fifth and sixth columns denote zero differences, and steps those aren't listed in the table have zero differences both for message words and chaining variables.

It is clear that the collision differential consists of two internal collisions respectively from 2-25 steps and 36-41 steps.

The sufficient conditions (Table 6) that ensure all the characteristics to hold can be easily verified by the properties of the Boolean functions given in Section 3. This further means that if M satisfies all the conditions in Table 6, M and M' consists of a collision.

The following is the derivation for the sufficient conditions in the step 9 of Table 5. The differential characteristic in step 9 is:

$$(b_2[-13, -14, 15], c_2[19, 20, -21, 22], d_2[14], a_2)$$
$$\longrightarrow (a_3[17], b_2[-13, -14, 15], c_2[19, 20, -21, 22], d_2[14])$$

1. According to (1) of Proposition 1, the conditions $c_{2,13} = d_{2,13}$ and $c_{2,15} = d_{2,15}$ ensure that the changes in 13-th and 15-th bits in b_2 result in no change in a_3.
2. According to (2) of Proposition 1, the conditions $b_{2,19} = 0$, $b_{2,20} = 0$, $b_{2,21} = 0$, and $b_{2,22} = 0$ ensure that the changes in 19-th, 20-th, 21-th and 22-th bits in c_2 result in no change in a_3.
3. From the property of function f, the conditions $b_{2,14} = 1$, $d_{2,14} = 0$ and $c_{2,14} = 0$ result in $f(b_{2,14}, c_{2,14}, d_{2,14}) = 0$ and $f(\neg b_{2,14}, c_{2,14}, \neg d_{2,14}) = 1$. So $\triangle a_3 = 2^{16}$.
4. The condition $a_{3,17} = 0$ ensures that $a'_3 = a_3[17]$.

Thus the above 10 conditions consists of a set of sufficient conditions for the differential characteristic in step 9.

4.2 Message Modification

From the conditions listed in Table 6, we know that the (M, M') is a collision with probability 2^{-122}. This is greatly lower than the birthday attack probability 2^{-64}. We can improve the probability to $2^{-6} \sim 2^{-2}$ by two types of message modification techniques, which we term as "single-step modification" and "multi-step modification."

Single-Step Modification. It is easy to modify M such that the conditions in round 1 hold. For example, m_1 can be modified as :

$$d_1 \leftarrow d_1 \oplus (d_{1,7} \lll 6) \oplus ((d_{1,8} \oplus a_{1,8}) \lll 7) \oplus ((d_{1,11} \oplus a_{1,11}) \lll 10)$$

$$m_1 \leftarrow (d_1 \ggg 7) - d_0 - F(a_1, b_0, c_0)$$

After simple-message modification, (M, M') is a collision with probability 2^{-25} by Table 6.

Multi-step Modification. Although the probability 2^{-25} is high enough for us to find many collisions of MD4, we also introduce a multi-message modification to correct the conditions in second round, and that greatly improves the probability. This modification technique is very important for analyzing other hash functions such as MD5, SHA-0, especially SHA-1.

The principle for multi-message modification is that the modifications for some messages consist of a partial collision in the first round which remains all the conditions in the first round to hold, but only change a bit of the second round. The details are as follows:

1. Modify m_0, m_1, m_2, m_3, m_4 successively by Table 1 to correct 5 conditions of a_5 in Table 6. For example, if $a_{5,19} = \overline{c_{4,19}}$, modify m_0, m_1, m_2, m_3, m_4 by Table 1 ($i = 19$). The changed message words don't change any condition of first round in Table 6, but correct $a_{5,19} = \overline{c_{4,19}}$ to $a_{5,19} = c_{4,19}$.
   It is noted that, the conditions in step 17 should be corrected from low bit to high bit, i.e. the order of the bits needed to be changed is:

$$a_{5,19} \rightarrow a_{5,26} \rightarrow a_{5,27} \rightarrow a_{5,29} \rightarrow a_{5,32}$$

2. Similarly, modify m_4, m_5, m_6, m_7, m_8 successively to correct 4 conditions of d_5.

$$d_{5,19} = a_{5,19},\ d_{5,26} = b_{4,26},\ d_{5,27} = b_{4,27},\ d_{5,29} = b_{4,29}$$

3. Utilize more precise modification to correct some other conditions. For example, we can use the internal collision in Table 2 in which there are three message words are changed to correct $c_{5,i}$, $i = 26, 27, 29, 32$. The precise modification should add some extra conditions in the first rounds (see Table 2) in advance. There are many other precise modifications. $c_{5,30}$ can be

Table 1. Message Modification for Correcting $a_{5,i}$, $i = 19, 26, 27, 29, 32$

			Modify m_i	Chaining values after message modification
1	m_0	3	$m_0 \longleftarrow m_0 \pm 2^{i-4}$	${a_1}^{new} = a_1[\pm i]$, b_0, c_0, d_0
2	m_1	7	$m_1 \longleftarrow (d_1 \ggg 7) - d_0 - f(a_1', b_0, c_0)$	d_1, ${a_1}^{new}$, b_0, c_0
3	m_2	11	$m_2 \longleftarrow (c_1 \ggg 11) - c_0 - f(d_1, a_1', b_0)$	c_1, d_1, ${a_1}^{new}$, b_0
4	m_3	19	$m_3 \longleftarrow (b_1 \ggg 19) - b_0 - f(c_1, d_1, a_1')$	b_1, c_1, d_1, ${a_1}^{new}$
5	m_4	3	$m_4 \longleftarrow (a_2 \ggg 3) - a_1' - f(b_1, c_1, d_1)$	a_2, b_1, c_1, d_1

Table 2. The Modification for Correcting $c_{5,i}$, $i = 26,\ 27,\ 29,\ 32$

				Modify m_i	Chaining values after message modification	The extra conditions in first round
6	d_2	m_5	7	$m_5 \leftarrow m_5 + 2^{i-17}$	$d_2[i-9]$, a_2, b_1, c_1	$d_{2,i-9} = 0$
7	c_2	m_6	11		c_2, $d_2[i-9]$, a_2, b_1	$a_{2,i-9} = b_{1,i-9}$
8	b_2	m_7	19		b_2, c_2, $d_2[i-9]$, a_2	$c_{2,i-9} = 0$
9	a_3	m_8	3	$m_8 \leftarrow m_8 - 2^{i-10}$	a_3, b_2, c_2, $d_2[i-9]$	$b_{2,i-9} = 0$
10	d_3	m_9	11	$m_9 \leftarrow m_9 - 2^{i-10}$	d_3, a_3, b_2, c_2	

Table 3. Two collisions for MD4. H is the hash value with little-endian and no message padding, and H^* is the hash value with big-endian and message padding

M_1	4d7a9c83 56cb927a b9d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f
	c69d71b3 f9e99198 d79f805e a63bb2e8 45dd8e31 97e31fe5 2794bf08 b9e8c3e9
M_1'	4d7a9c83 d6cb927a 29d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f
	c69d71b3 f9e99198 d79f805e a63bb2e8 45dc8e31 97e31fe5 2794bf08 b9e8c3e9
H	5f5c1a0d 71b36046 1b5435da 9b0d807a
H^*	4d7e6a1d efa93d2d de05b45d 864c429b
M_2	4d7a9c83 56cb927a b9d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f
	c69d71b3 f9e99198 d79f805e a63bb2e8 45dd8e31 97e31fe5 f713c240 a7b8cf69
M_2'	4d7a9c83 d6cb927a 29d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f
	c69d71b3 f9e99198 d79f805e a63bb2e8 45dc8e31 97e31fe5 f713c240 a7b8cf69
H	e0f76122 c429c56c ebb5e256 b809793
H^*	c6f3b3fe 1f4833e0 697340fb 214fb9ea

corrected by other modification. By various modifications, besides two conditions in the third round, almost all the conditions in rounds 1-2 will be corrected. The probability can be among $2^{-6} \sim 2^{-2}$.

The complexity of finding a collision doesn't exceed 2^8 MD4 computations. To select a message M is only to change the last two words from the previous selected message M. So, finding (M, M') only needs about one-time single-message modification for the first 14 words. This time can be neglected. For each selected message M, it is only needs two-time single-message modifications for the last two words and about 20 -time advanced modifications for correcting 20 conditions in the second round, and each multi-message modification only needs about

a few step operations, so the total time for both kinds of modifications is about two MD4 computations for each selected message. According to the probability of the collision differential, it is easy to know that the complexity of finding (M, M') does not exceed 2^8 MD4 computations. We give two collisions for MD4 in the Table 3.

5 The Collision Attack on RIPEMD

We select a collision differential for RIPEMD as follows:

$$\Delta H_0 = 0 \overset{(M,M')}{\longrightarrow} \Delta H = 0$$

such that

$$m'_3 = m_3 + 2^{20},\ m'_{10} = m_{10} + 2^{18} + 2^{31},\ m'_{15} = m_{15} + 2^{31},$$

$$m'_i = m_i,\ i \neq 3, 10, 15.$$

The reason for the choice of M' is that M and M' can easily collide in round 3 with probability 2^{-4}.

The differential characteristics and sufficient conditions can be referred to Table 7 and Table 8.

The following mainly describes the message modification for RIPEMD. Because RIPEMD has two copies of MD4, the modification is more complicated than that of MD4.

Message Modification for Correcting Conditions in the First Round. Select M, we make the modification for M word by word so that both copies with the modified M satisfy the conditions in the first round.

1. Modify m_{i-1} such that i-th step conditions in the left copy hold. The modification is the same as the single-message modification in Section 4.
2. Correct the conditions in the right copy from low bit to high bit. There are many kinds of modifications. The following gives two kinds of modification techniques.
   For example, we correct $aaa_{i,j} = 0$ to $aaa_{i,j} = 1$ by the following methods.
   (a) Correct the condition by bit carry. If $j-1$-bit has no constraint condition in table 8, and $aa_{i,j-1} = \overline{aa_{i,j-1}}$, let

   $$m_i \leftarrow m_i \pm 2^{j-2-s_i}.$$

   We select the modification which results in bit carry in the right and no carry in the left.
   (b) Correct the condition by changing $(j - s_i) - th$ bit of chaining variables in the nonlinear round function ψ.
      i. Change $(j - s_i) - th$ bit of some chaining variables in the nonlinear round function F by modifying a previous message word, such that the changed bit doesn't occur in Table 8, and the changed bit only causes one of $aa_{i,j}$ and $aaa_{i,j}$ changes.

ii. If $aa_{i,j} = aaa_{i,j} = 0$, modify the next bit of aaa_i.
iii. If $aa_{i,j} = aaa_{i,j} = 1$, let

$$m_i \leftarrow m_i - 2^{j-1-s_i},$$

then modify the next bit of aaa_i.

By combining the above two methods, we can get some other methods to correct $aaa_{i,j}$. For example, if $j-1$-bit has no constraint condition in table 8, and $aa_{i,j-1} = aa_{i,j-1}$, the bit-carry correction of (a) isn't available. We can use (b) or the lower bit carry to change $aa_{i,j-1}$ or $aaa_{i,j-1}$ such that $aa_{i,j-1} = \overline{aaa_{i,j-1}}$, and then use the bit carry.

Remarks. For RIPEMD, a non-zero differential in the first round is an impossible differential with a very high probability. The reason that results in the phenomenon is that, the conditions of both copies in some step cannot hold simultaneously. Among 30 collision differentials we selected, only one can produce the real collisions.

Message Modification for Correcting Some Second Round Conditions. By the multi-message modification in Section 4 to correct the conditions of left copy in the second round. There are about 16 conditions are left, so the modified M and M' is a collision with probability 2^{-16}, and the complexity is about 2^{18} RIPEMD computations. Two collisions for RIPEMD can be seen in Table 4.

Table 4. Two collisions for RIPEMD. H is the hash value with little-endian and no message padding, and H^* is the hash value with big-endian and message padding

M_1	579faf8e 9ecf579 574a6aba 78413511 a2b410a4 ad2f6c9f b56202c 4d757911
	bdeaae7 78bc91f2 47bc6d7d 9abdd1b1 a45d2015 817104ff 264758a8 61064ea5
M_1'	579faf8e 9ecf579 574a6aba 78513511 a2b410a4 ad2f6c9f b56202c 4d757911
	bdeaae7 78bc91f2 c7c06d7d 9abdd1b1 a45d2015 817104ff 264758a8 e1064ea5
H	1fab152 1654a31b 7a33776a 9e968ba7
H^*	dd6478dd 9a7d821c aa018648 e5e792e9
M_2	579faf8e 9ecf579 574a6aba 78413511 a2b410a4 ad2f6c9f b56202c 4d757911
	bdeaae7 78bc91f2 47bc6d7d 9abdd1b1 a45d2015 a0a504ff b18d58a8 e70c66b6
M_2'	579faf8e 9ecf579 574a6aba 78513511 a2b410a4 ad2f6c9f b56202c 4d757911
	bdeaae7 78bc91f2 c7c06d7d 9abdd1b1 a45d2015 a0a504ff b18d58a8 670c66b6
H	1f2c159f 569b31a6 dfcaa51a 25665d24
H^*	88cea096 c773c29f 04cd9698 4a41d139

6 Theoretical Pre-image Attack on MD4

For a secure hash function, there are two important security properties, one property is collision-resistance, another is one-wayness which is to find a second

pre-image or a pre-image. In this section, we will show that we can give a second pre-image attack on MD4 for a set of weak messages.

For a hash function with l-bit hash value, it's ideal security strength against the second pre-image attack is that, for any message M, to find another message M' such that $h(M) = h(M')$ is not higher than the exhaustive search probability of 2^{-l}.

Theorem 1 (Second Pre-image Attack for Weak Messages). *For a weak message, we can find another message such that these two different messages produce the same hash code. The complexity is only one-time MD4 computation. A random selected message is weak with probability* 2^{-122}.

Proof. For any message M, we select M' such that

$$M' = M + \triangle M$$

$$\triangle M = M' - M = (\triangle m_0, \triangle m_1, ..., \triangle m_{15})$$

$$\triangle m_1 = 2^{31}, \triangle m_2 = 2^{31} - 2^{28}, \triangle m_{12} = -2^{16},$$

$$\triangle m_i = 0, 0 \leq i \leq 15, i \neq 1, 2, 12.$$

From the conditions in Table 6, we know that, if M satisfies all the 122 conditions, M' is the second pre-image of $h(M)$.

There are $2^{512}/2^{122} = 2^{391}$ one-block messages satisfy all the conditions. This completes the proof. □

Any message M can be modified with the techniques in Section 4 so that almost all the conditions in rounds 1-2 hold. For the resulting message, say M', we then find a second pre-image M'' of $h(M')$ with probability 2^{-2} to 2^{-6}. This fact can be interpreted as a chosen-message 2nd pre-image attack, since M' is not chosen freely but "close" to M. One message "close" to other message implies that the hamming weight of the difference for two messages is low. For example, given any random message M, if we only fulfil the the single-message modification, the chosen message M' is the 2nd pre-image of other message M'' with probability 2^{23} (excluding two conditions in 17-step). According to the conditions in Table 6, we can get M' by modifying M about 50 bits, so the difference hamming weight for two messages is 50 on average. When applying the multi-message modification, although the probability can be improved to 2^{-2} to 2^{-6}, the hamming weight may be greatly increased. The best method is to fulfil a kind of precise message modification, and correct a condition by increasing about 3 hamming weights. So, the difference hamming weight can be controlled within 110 on average.

7 Conclusion

In this paper, we have presented efficient collision search attacks on MD4 and RIPEMD. We have shown that only about 4 to 64 random selected messages are

needed in order to find a collision of MD4, and only about 2^{16} random selected messages to for RIPEMD.

We have introduced three important analytical techniques that are very important for the effectiveness of the attacks:

1. How to find an efficient differential that is composed of one collision.
2. Determine all the conditions under which the collision happens.
3. For any message M, make some modification to M to guarantee that almost all the conditions hold.

Remarks. Our collision search attack on MD4 implies that for a weak message a 2nd pre-image can be found with complexity below 2^8. The probability for a random messages to be weak with respect to MD4 is 2^{-122}. However, this can be improved significantly. In fact note that Theorem 1 directly come from the collision differential path in Section 4, where the differential path is chosen to minimize the complexity of our collision attack. Hence it is not optimal for our pre-image attack. The number of weak messages is determined by the number of conditions specified in Table 1. By finding other differential paths with the least number of conditions, we believe that the probability of weak messages can be increased significantly. In fact, the latest 2nd pre-image attack can be improved to 2^{-72} which is found by Hongbo Yu Gaoli Wang et al.

We also note that for SHA-0, given any random message, it is a weak message with about probability 2^{-107} which is a surprising result compared to the exhaustive search probability 2^{-160}.

Acknowledgements

It is a pleasure to acknowledge Hans Dobbertin, Magnus Daum for their important advice, corrections, and suggestions, and for spending their precious time on our research.

Xiaoyun Wang's research is supported by the National Natural Science Foundation of China (Grant No. 90304009). Dengguo Feng's research is supported by 973 project (Grant No. G19990358).

References

1. E. Biham, A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.
2. E. Biham, R. Chen, *Near collision for SHA-0,* Advances in Cryptology, Crypto'04, 2004, LNCS 3152, pp. 290-305.
3. B. den Boer, A. Bosselaers, *Collisions for the compression function of MD5,* Advances in Cryptology, Eurocrypt'93.
4. F. Chaband, A. Joux, Differential Collisions in SHA-0, Advances in Cryptology, Crypto'98 Proceedings, 1998.

5. H. Dobbertin, Cryptanalysis of MD4, Fast Software Encryption, LNCS 1039, D. Gollmann, Ed., Springer-Verlag, 1996.
6. H. Dobbertin, Cryptanaltysis of MD5 Compress, Presented at the Rump Session of Eurocrypt'96.
7. H. Dobbertin, RIPEMD with Two Round Compress Function Is Not Collision-Free, Journal of Cryptology(1997) 10:51-69, 1997.
8. H. Dobbertin, The First Two Rounds of MD4 are Not One-Way, Fast Software Encryption, 1998.
9. H. Dobbertin, A. Bosselaers, B. Preneel, RIPMEMD-160:A Strengthened Version of RIPMMD, Fast Software Encryption, LNCS 1039, 1996.
10. FIPS 180-1, Secure hash standard, NIST, US Department of Commerce, Washington D. C., April 1995. Springer-Verlag, 1996.
11. FIPS 180-2, Secure Hash Standard, http://csrc.nist.gov/publications/,2002.
12. Joux, A., Collisions for SHA-0, Rump Session of CRYPTO'04, 2004.
13. RIPE,Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evalutiobn(RIPE-RACE 1040), LNCS 1007, 1995.
14. R. L., Rivest, The MD4 Message Digest Algorithm, Crypo'90 Proceedings, 1991.
15. R. L. Rivest, The MD5 Message-Digest Algorithm, Request for Comments (RFC 1320), Internet Activities Board, Internet Privacy Task Force, April 1992.
16. Bart Van Rompay, A. Biryukov, B. Preneel, J. Vandewalle, Cryptanalysis of 3-pass HAVAL, Asiacrypto'03 Proceedings, pp. 228-245, 2003.
17. X.Y. Wang, F.D. Guo, X.J. Lai, H.B. Yu, Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, Rump Session of Crypto'04, E-print, 2004.
18. Y. Zheng, J. Pieprzyk, J. Seberry, HAVAL–A One-way Hashing Algorithm with Variable Length of Output, Auscrypto'92 Proceedings, pp.83-104.

Appendix

In the appendix, we give the tables for the differential paths and the set of sufficient conditions that are used in our collision search attacks on MD4 and RIPEMD.

Table 5. Differential Characteristics in the Collision Differential for MD4

Step	Chaining value for M	$w_{j,i}$	Shift	Δm_i	The i-th step difference	The i-th output for M'
1	a_1	m_0	3			a_1
2	d_1	m_1	7	2^{31}	2^6	$d_1[7]$
3	c_1	m_2	11	$-2^{28}+2^{31}$	-2^7+2^{10}	$c_1[-8,11]$
4	b_1	m_3	19		2^{25}	$b_1[26]$
5	a_2	m_4	3			a_2
6	d_2	m_5	7		2^{13}	$d_2[14]$
7	c_2	m_6	11		$-2^{18}+2^{21}$	$c_2[19,20,-21,22]$
8	b_2	m_7	19		2^{12}	$b_2[-13,-14,15]$
9	a_3	m_8	3		2^{16}	$a_3[17]$
10	d_3	m_9	7		$2^{19}+2^{20}-2^{25}$	$d_3[20,-21,-22,23,-26]$
11	c_3	m_{10}	11		-2^{29}	$c_3[-30]$
12	b_3	m_{11}	19		2^{31}	$b_3[32]$
13	a_4	m_{12}	3	-2^{16}	$2^{22}+2^{25}$	$a_4[23,26]$
14	d_4	m_{13}	7		$-2^{26}+2^{28}$	$d_4[-27,-29,30]$
15	c_4	m_{14}	11			c_4
16	b_4	m_{15}	19		2^{18}	$b_4[19]$
17	a_5	m_0	3		$2^{25}-2^{28}-2^{31}$	$a_5[-26,27,-29,-32]$
18	d_5	m_4	5			d_5
19	c_5	m_8	9			c_5
20	b_5	m_{12}	13	-2^{16}	$-2^{29}+2^{31}$	$b_5[-30,32]$
21	a_6	m_1	3	2^{31}	$2^{28}-2^{31}$	$a_6[-29,30,-32]$
22	d_6	m_5	5			d_6
23	c_6	m_9	9			c_6
24	b_6	m_{13}	13			b_6
25	a_7	m_2	3	$-2^{28}+2^{31}$		a_7
...	...					.....
36	b_9	m_{12}	15	-2^{16}	2^{31}	$b_9[-32]$
37	a_{10}	m_2	3	$-2^{28}+2^{31}$	2^{31}	$a_{10}[-32]$
38	d_{10}	m_{10}	9			d_{10}
39	c_{10}	m_6	11			c_{10}
40	b_{10}	m_{14}	15			b_{10}
41	a_{11}	m_1	3	2^{31}		a_{11}

Table 6. A Set of Sufficient Conditions for Collisions of MD4

a_1	$a_{1,7} = b_{0,7}$
d_1	$d_{1,7} = 0$, $d_{1,8} = a_{1,8}$, $d_{1,11} = a_{1,11}$
c_1	$c_{1,7} = 1$, $c_{1,8} = 1$, $c_{1,11} = 0$, $c_{1,26} = d_{1,26}$
b_1	$b_{1,7} = 1$, $b_{1,8} = 0$, $b_{1,11} = 0$, $b_{1,26} = 0$
a_2	$a_{2,8} = 1$, $a_{2,11} = 1$, $a_{2,26} = 0$, $a_{2,14} = b_{1,14}$
d_2	$d_{2,14} = 0$, $d_{2,19} = a_{2,19}$, $d_{2,20} = a_{2,20}$, $d_{2,21} = a_{2,21}$, $d_{2,22} = a_{2,22}$, $d_{2,26} = 1$
c_2	$c_{2,13} = d_{2,13}$, $c_{2,14} = 0$, $c_{2,15} = d_{2,15}$, $c_{2,19} = 0$, $c_{2,20} = 0$, $c_{2,21} = 1$, $c_{2,22} = 0$
b_2	$b_{2,13} = 1$, $b_{2,14} = 1$, $b_{2,15} = 0$, $b_{2,17} = c_{2,17}$, $b_{2,19} = 0$, $b_{2,20} = 0$, $b_{2,21} = 0$ $b_{2,22} = 0$
a_3	$a_{3,13} = 1$, $a_{3,14} = 1$, $a_{3,15} = 1$, $a_{3,17} = 0$, $a_{3,19} = 0$, $a_{3,20} = 0$, $a_{3,21} = 0$, $a_{3,23} = b_{2,23}$ $a_{3,22} = 1$, $a_{3,26} = b_{2,26}$
d_3	$d_{3,13} = 1$, $d_{3,14} = 1$, $d_{3,15} = 1$, $d_{3,17} = 0$, $d_{3,20} = 0$, $d_{3,21} = 1$, $d_{3,22} = 1$, $d_{3,23} = 0$, $d_{3,26} = 1$, $d_{3,30} = a_{3,30}$
c_3	$c_{3,17} = 1$, $c_{3,20} = 0$, $c_{3,21} = 0$, $c_{3,22} = 0$, $c_{3,23} = 0, c_{3,26} = 0, c_{3,30} = 1, c_{3,32} = d_{3,32}$
b_3	$b_{3,20} = 0$, $b_{3,21} = 1$, $b_{3,22} = 1$, $b_{3,23} = c_{3,23}$, $b_{3,26} = 1$, $b_{3,30} = 0$, $b_{3,32} = 0$
a_4	$a_{4,23} = 0$, $a_{4,26} = 0$, $a_{4,27} = b_{3,27}$, $a_{4,29} = b_{3,29}$, $a_{4,30} = 1$, $a_{4,32} = 0$
d_4	$d_{4,23} = 0$, $d_{4,26} = 0$, $d_{4,27} = 1$, $d_{4,29} = 1$, $d_{4,30} = 0$, $d_{4,32} = 1$
c_4	$c_{4,19} = d_{4,19}$, $c_{4,23} = 1$, $c_{4,26} = 1$, $c_{4,27} = 0$, $c_{4,29} = 0$, $c_{4,30} = 0$
b_4	$b_{4,19} = 0$, $b_{4,26} = c_{4,26} = 1$, $b_{4,27} = 1$, $b_{4,29} = 1$, $b_{4,30} = 0$
a_5	$a_{5,19} = c_{4,19}$, $a_{5,26} = 1$, $a_{5,27} = 0$, $a_{5,29} = 1$, $a_{5,32} = 1$
d_5	$d_{5,19} = a_{5,19}$, $d_{5,26} = b_{4,26}$, $d_{5,27} = b_{4,27}$, $d_{5,29} = b_{4,29}$, $d_{5,32} = b_{4,32}$
c_5	$c_{5,26} = d_{5,26}$, $c_{5,27} = d_{5,27}$, $c_{5,29} = d_{5,29}$, $c_{5,30} = d_{5,30}$, $c_{5,32} = d_{5,32}$
b_5	$b_{5,29} = c_{5,29}$, $b_{5,30} = 1$, $b_{5,32} = 0$
a_6	$a_{6,29} = 1$, $a_{6,32} = 1$
d_6	$d_{6,29} = b_{5,29}$
c_6	$c_{6,29} = d_{6,29}$, $c_{6,30} = d_{6,30} + 1$, $c_{6,32} = d_{6,32} + 1$
b_9	$b_{9,32} = 1$
a_{10}	$a_{10,32} = 1$

Table 7. Differential Characteristics in a Collision Differential of RIPEMD

Step	Chaining value for M	$w_{j,i}$	Shift	Δm_i	The i-th step difference	The i-th output for M'
0	a_1	m_0	11			a_1
1	d_1	m_1	14			d_1
2	c_1	m_2	15			c_1
3	b_1	m_3	12	2^{20}	1	$b_1[-1,-2,-3,4]$
4	a_2	m_4	5		2^6	$a_2[7]$
5	d_2	m_5	8		2^9-2^{11}	$d_2[10,-12]$
6	c_2	m_6	7		$2^{16}-2^{18}$	$c_2[17,-19]$
7	b_2	m_7	9		$2^9+2^{25}+2^{27}$	$b_2[10,-26,27,28]$
8	a_3	m_8	11		-2^5+2^{17}	$a_3[-6,18]$
9	d_3	m_9	13		-2^{23}	$d_3[24,25,-26]$
10	c_3	m_{10}	14	$2^{18}+2^{31}$	$-2^{13}+2^{30}$	$c_3[-14,31]$
11	b_3	m_{11}	15		$2^{10}+2^{24}$	$b_3[11,25]$
12	a_4	m_{12}	6		$-2^{11}+2^{23}$	$a_4[12,13,-14,24]$
13	d_4	m_{13}	7			d_4
14	c_4	m_{14}	9		2^7-2^{23}	$c_4[8,24,-25]$
15	b_4	m_{15}	8	2^{31}	-2^7+2^{18}	$b_4[-8,19]$
16	a_5	m_7	7		-2^{18}	$a_5[-19]$
17	d_5	m_4	6			d_5
18	c_5	m_{13}	8		2^{31}	$c_5[-32]$
19	b_5	m_1	13		-2^{20}	b_5[-21]
20	a_6	m_{10}	11	$2^{18}+2^{31}$		a_6
21	d_6	m_6	9			d_6
22	c_6	m_{15}	7	2^{31}		c_6
23	b_6	m_3	15	2^{20}		b_6
24	a_7	m_{12}	7			a_7
25	d_7	m_0	12			d_7
26	c_7	m_9	15			c_7
27	b_7	m_5	9			b_7
28	a_8	m_{14}	7			a_8
29	d_8	m_2	11			d_8
30	c_8	m_{11}	13			c_8
31	b_8	m_8	12			b_8
32	a_9	m_3	11	2^{20}	2^{31}	$a_9[32]$
33	d_9	m_{10}	13	$2^{18}+2^{31}$	2^{31}	$d_9[32]$
34	c_9	m_2	14			c_9
35	b_9	m_4	7			b_9
36	a_{10}	m_9	14			a_{10}
37	d_{10}	m_{15}	9	2^{31}		d_{10}

Table 8. A Set of Sufficient Conditions for Collisions of RIPEMD

a_1	
d_1	$d_{1,2} = 1$
c_1	$c_{1,1} = d_{1,1}$, $c_{1,2} = 0$, $c_{1,3} = d_{1,3}$, $c_{1,4} = d_{1,4}$
b_1	$b_{1,1} = 1$, $b_{1,2} = 1$, $b_{1,3} = 1$, $b_{1,4} = 0$, $b_{1,7} = c_{1,7}$
a_2	$a_{2,7=0}$, $a_{2,1} = 0$, $a_{2,2} = 1$, $a_{2,3} = 1$, $a_{2,4} = 0$, $a_{2,10} = \overline{b_{1,10}} = 1$, $a_{2,12} = \overline{b_{1,12}} = 1$ $a_{2,17} = 0$
d_2	$d_{2,1} = 1$, $d_{2,2} = 1$, $d_{2,3} = 1$, $d_{2,4} = 1$, $d_{2,7} = 0$, $d_{2,10} = 0$, $d_{2,12} = 1$, $d_{2,17} = 1$, $d_{2,19} = \overline{a_{2,19}} = 0$
c_2	$c_{2,17} = 0$, $c_{2,19} = 1$, $c_{2,10} = 0$, $c_{2,7} = 1$, $c_{2,12} = 0$, $c_{2,26} = d_{2,26}$, $c_{2,27} = \overline{d_{2,27}} = 0$, $c_{2,28} = d_{2,28}$
b_2	$b_{2,6} = c_{2,6}$, $b_{2,10} = 0$, $b_{2,12} = 1$, $b_{2,17} = 0$, $b_{2,18} = c_{2,18}$, $b_{2,19} = 0$, $b_{2,26} = 1$, $b_{2,27} = 0$, $b_{2,28} = 0$
a_3	$a_{3,6} = 1$, $a_{3,10} = 1$, $a_{3,17} = 1$, $a_{3,18} = 0$,$a_{3,19} = 1$, $a_{3,24} = b_{2,24}$, $a_{3,25} = b_{2,25}$, $a_{3,26} = 0$, $a_{3,27} = 0$, $a_{3,28} = 0$
d_3	$d_{3,6} = 0$, $d_{3,10} = 1, d_{3,14} = a_{3,14}, d_{3,18} = 0, d_{3,24} = 0, d_{3,25} = 0, d_{3,26} = 1, d_{3,27} = 1$, $d_{3,28} = 1$, $d_{3,31} = a_{3,31}$
c_3	$c_{3,6} = 1$, $c_{3,11} = d_{3,11}$, $c_{3,14} = 1$, $c_{3,18} = 1$, $c_{3,24} = 0$, $c_{3,25} = 0$, $c_{3,26} = 1$, $c_{3,31} = 0$
b_3	$b_{3,11} = 0, b_{3,12} = c_{3,12}, b_{3,13} = c_{3,13}, b_{3,14} = 0, b_{3,24} = 1, b_{3,25} = 0, b_{3,26} = 1, b_{3,31} = 0$
a_4	$a_{4,11} = 0$, $a_{4,12} = 0$, $a_{4,13} = 0$, $a_{4,14} = 1$, $a_{4,24} = 0$, $a_{4,25} = 0$, $a_{4,31} = 1$
d_4	$d_{4,8} = a_{4,8}$, $d_{4,11} = 1$, $d_{4,12} = 0$, $d_{4,13} = 0$, $d_{4,14} = 1$, $d_{4,24} = 0$, $d_{4,25} = 1$,
c_4	$c_{4,8} = 0$, $c_{4,12} = 1$, $c_{4,13} = 1$, $c_{4,14} = 1$, $c_{4,19} = d_{4,19}$, $c_{4,24} = 0$, $c_{4,25} = 1$,
b_4	$b_{4,8} = 1$, $b_{4,19} = 0$, $b_{4,24} = \overline{d_{4,24}}$, $b_{4,25} = \overline{d_{4,25}}$
a_5	$a_{5,19} = 1$, $a_{5,24} = b_{4,24}$, $a_{5,25} = b_{4,25}$
d_5	$d_{5,8} = \overline{a_{5,8}}$, $d_{5,32} = a_{5,32}$
c_5	$c_{5,19} = \overline{d_{5,19}}$, $c_{5,21} = d_{5,21}$, $c_{5,32} = 1$
b_5	$b_{5,21} = 1$, $b_{5,32} = \overline{d_{5,32}}$
a_6	$a_{6,21} = c_{5,21}$, $a_{6,32} = b_{5,32}$
d_6	$d_{6,21} = a_{6,21}$
a_9	$a_{9,32} = 0$,
d_9	$d_{9,32} = 0$
a_{10}	

How to Break MD5 and Other Hash Functions

Xiaoyun Wang and Hongbo Yu

Shandong University, Jinan 250100, China
xywang@sdu.edu.cn
yhb@mail.sdu.edu.cn

Abstract. MD5 is one of the most widely used cryptographic hash functions nowadays. It was designed in 1992 as an improvement of MD4, and its security was widely studied since then by several authors. The best known result so far was a semi free-start collision, in which the initial value of the hash function is replaced by a non-standard value, which is the result of the attack. In this paper we present a new powerful attack on MD5 which allows us to find collisions efficiently. We used this attack to find collisions of MD5 in about 15 minutes up to an hour computation time. The attack is a differential attack, which unlike most differential attacks, does not use the exclusive-or as a measure of difference, but instead uses modular integer subtraction as the measure. We call this kind of differential a *modular differential*. An application of this attack to MD4 can find a collision in less than a fraction of a second. This attack is also applicable to other hash functions, such as RIPEMD and HAVAL.

1 Introduction

People know that digital signatures are very important in information security. The security of digital signatures depends on the cryptographic strength of the underlying hash functions. Hash functions also have many other applications in cryptography such as data integrity, group signature, e-cash and many other cryptographic protocols. The use of hash functions in these applications not only ensure the security, but also greatly improve the efficiency. Nowadays, there are two widely used hash functions – MD5 [18] and SHA-1 [12].

MD5 is a hash function designed by Ron Rivest as a strengthened version of MD4 [17]. Since its publication, some weaknesses has been found. In 1993, B. den Boer and A. Bosselaers [3] found a kind of pseudo-collision for MD5 which consists of the same message with two different sets of initial values. This attack discloses the weak avalanche in the most significant bit for all the chaining variables in MD5. In the rump session of Eurocrypt'96, H. Dobbertin [8] presented a semi free-start collision which consists of two different 512-bit messages with a chosen initial value IV_0'.

$$a_0 = \texttt{0x12ac2375},\ b_0 = \texttt{0x3b341042},\ c_0 = \texttt{0x5f62b97c},\ d_0 = \texttt{0x4ba763ed}$$

A general description of this attack was published in [9].

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 19–35, 2005.

Although H. Dobbertin cannot provide a real collision of MD5, his attack reveals the weak avalanche for the full MD5. This provides a possibility to find a special differential with one iteration.

In this paper we present a new powerful attack that can efficiently find a collision of MD5. From H. Dobbertin's attack, we were motivated to study whether it is possible to find a pair of messages, each consists of two blocks, that produce collisions after the second block. More specifically, we want to find a pair (M_0, M_1) and (M'_0, M'_1) such that

$$\begin{aligned}(a, b, c, d) &= \mathrm{MD5}(a_0, b_0, c_0, d_0, M_0),\\ (a', b', c', d') &= \mathrm{MD5}(a_0, b_0, c_0, d_0, M'_0),\\ \mathrm{MD5}(a, b, c, d, M_1) &= \mathrm{MD5}(a', b', c', d', M'_1),\end{aligned}$$

where a_0, b_0, c_0, d_0 are the initial values for MD5. We show that such collisions of MD5 can be found efficiently, where finding the first blocks (M_0, M'_0) takes about 2^{39} MD5 operations, and finding the second blocks (M_1, M'_1) takes about 2^{32} MD5 operations. The application of this attack on IBM P690 takes about an hour to find M_0 and ${M_0}'$, where in the fastest cases it takes only 15 minutes. Then, it takes only between 15 seconds to 5 minutes to find the second blocks M_1 and ${M_1}'$. Two such collisions of MD5 were made public in the Crypto'04 rump session [19].

This attack is applicable to many other hash functions as well, including MD4, HAVAL-128 and RIPEMD ([17], [20], [15]). In the case of MD4, the attack can find a collision within less than a second, and can also find second pre-images for many messages.

In Crypto'04 Eli Biham and Rafi Chen presented a near-collision attack on SHA-0 [2], which follows the lines of the technique of [4]. In the rump session they described their new (and improved) results on SHA-0 and SHA-1 (including a multi-block technique and collisions of reduced SHA-1). Then, A.J̃oux presented a 4-block full collision of SHA-0 [14], which is a further improvement of these results. Both these works were made independently of this paper.

This paper is organized as follows: In Section 2 we briefly describe MD5. Then in Section 3 we give the main ideas of our attack, and in Section 4 we give a detailed description of the attack. Finally, in Section 5 we summarize the paper, and discuss the applicability of this attack to other hash functions.

2 Description of MD5

In order to conveniently describe the general structure of MD5, we first recall the iteration process for hash functions.

Generally a hash function is iterated by a compression function $X = f(Z)$ which compresses l-bit message block Z to s-bit hash value X where $l > s$. For MD5, $l = 512$, and $s = 128$. The iterating method is usually called the Merkle-Damgard meta-method (see [6], [16]). For a padded message M with multiples of l-bit length, the iterating process is as follows:

$$H_{i+1} = f(H_i, M_i), \quad 0 \le i \le t - 1.$$

Here $M = (M_0, M_2, \cdots, M_{t-1})$, and $H_0 = IV_0$ is the initial value for the hash function.

In the above iterating process, we omit the padding method because it has no influence on our attack.

The following is to describe the compression function for MD5. For each 512-bit block M_i of the padded message M, divide M_i into 32-bit words, $M_i = (m_0, m_1, ..., m_{15})$. The compression algorithm for M_i has four rounds, and each round has 16 operations. Four successive step operations are as follows:

$$\begin{aligned}
a &= b + ((a + \phi_i(b,c,d) + w_i + t_i) \lll s_i),\\
d &= a + ((d + \phi_{i+1}(a,b,c) + w_{i+1} + t_{i+1}) \lll s_{i+1}),\\
c &= d + ((c + \phi_{i+2}(d,a,b) + w_{i+2} + t_{i+2}) \lll s_{i+2}),\\
b &= c + ((b + \phi_{i+3}(c,d,a) + w_{i+3} + t_{i+3}) \lll s_{i+3}),
\end{aligned}$$

where the operation + means ADD modulo 2^{32}. t_{i+j} and s_{i+j} $(j = 0,\ 1,\ 2,\ 3)$ are step-dependent constants. w_{i+j} is a message word. $\lll s_{i+j}$ is circularly left-shift by s_{i+j} bit positions. The details of the message order and shift positions can be seen in Table 3.

Each round employs one nonlinear round function, which is given below.

$$\begin{aligned}
\Phi_i(X,Y,Z) &= (X \wedge Y) \vee (\neg X \wedge Z), && 0 \le i \le 15,\\
\Phi_i(X,Y,Z) &= (X \wedge Z) \vee (Y \wedge \neg Z), && 16 \le i \le 31,\\
\Phi_i(X,Y,Z) &= X \oplus Y \oplus Z, && 32 \le i \le 47,\\
\Phi_i(X,Y,Z) &= Y \oplus (X \vee \neg Z), && 48 \le i \le 63,
\end{aligned}$$

where X, Y, Z are 32-bit words.

The chaining variables are initialized as:

$$a = \texttt{0x67452301},\ b = \texttt{0xefcdab89},\ c = \texttt{0x98badcfe},\ d = \texttt{0x10325476}.$$

We select a collision differential with two iterations as follows: Let $H_{i-1} = (aa, bb, cc, dd)$ be the chaining values for the previous message block. After four rounds, the compression value H_i is obtained by wordwise addition of the chaining variables to H_{i-1}.

3 Differential Attack for Hash Functions

3.1 The Modular Differential and the XOR Differential

The most important analysis method for hash functions is differential attack which is also one of most important methods for analyzing block ciphers. In general, the differential attack especially in block ciphers is a kind of XOR differential attack which uses exclusive-or as the difference. The differential attack was introduced by E. Biham and A. Shamir to analyze the security of DES-like cryptosystems. E. Biham and A. Shamir [1], described that differential cryptanalysis is a method which analyzes the effect of particular differences in plain text pairs on the differences of the resultant cipher text pairs.

The differential definition in this paper is a kind of precise differential which uses the difference in term of integer modular subtraction. A similar definition about the differential with the integer subtraction as the measure of difference were described in [5] for differential analysis of RC6.

We also use modular characteristics, which describe for each round with both the differences in term of integer modular subtraction and the differences in term of XOR. The combination of both kinds of differences give us more information than each of them keep by itself. For example, when the modular integer subtraction difference is $X' - X = 2^6$ for some value X, the XOR difference $X' \oplus X$ can have many possibilities, which are

1. One-bit difference in bit 7, i.e., `0x00000040`. In this case $X' - X = 2^6$ which means that bit 7 in X' is 1 and bit 7 in X is 0.
2. Two-bit difference, in which a different carry is transferred from bit 7 to bit 8, i.e., `0x000000C0`. In this case $X' - X = 2^6$, but the carry to bit 8 is different in X and X', so X'_7 is now 0, and $X_7 = 1$, while $X'_8 = 1$, and $X_8 = 0$. (i.e., bits 7 and 8 together in X' are 10 in binary, and in X there are 01 in binary).
3. Three-bit difference, in which a different carry is transferred from bit 7 to bit 8 and then to bit 9, i.e., `0x000001C0`. In this case bits 7, 8, and 9 in X' are 0, 0, and 1, respectively, and in X they are the complement of these values.
4. Similarly, there can be more carries to further bits, and the binary form of X' is $1000\ldots$, and of X is $0111\ldots$.
5. In case the former difference is negative, the XOR differences still look the same, but the values of X and X' are exchanged (i.e., X is of the form $1000\ldots$, and X' of the form $0111\ldots$).

In order to explain our attack clearly, we refer to the modular differences in the differential path (see Table 3) with both kinds of differences together, i.e., the difference is marked as a positive or a negative integer (modulo 2^{32}) and also with the XOR difference. But then the XOR difference is marked by the list of active bits with their relative sign, i.e., in the list of bits, the bits whose value in X is zero are marked without a sign, and the values whose value in X is 1 are marked with a negative sign. For example, the difference $-2^6, [7, 8, 9, \ldots, 22, -23]$ marks the integer modular subtraction difference $X' - X = -2^6$ (with $X' < X$), with many carries which start from bit 7 up to bit 23. All bits of X from bit 7 to bit 22 are 0, and bit 23 is 1, while all bits of X' from bit 7 to bit 22 are 1, and bit 23 is 0. A more complicated example is $-1 - 2^6 + 2^{23} - 2^{27}, [1, 2, 3, 4, 5, -6, 7, 8, 9, 10, 11, -12, -24, -25, -26, 27, 28, 29, 30, 31, -32]$, where the integer modular subtraction difference is composed of several (positive and negative) exponents of 2, and the XOR difference has many difference due to carries. Note that when the carry arrives to bit 32, a further (dropped) carry may happen, and then there is no negative sign in bit 32.

It should be noted that the modular differential has been used earlier to analyze some hash functions ([4], [7], [10]). Compared with these attacks, our attack has the following advantages:

1. Our attack is to find collisions with two iterations, i. e., each message in the collision includes two message blocks (1024-bit).
2. Our attack is a precise differential attack in which the characteristics are more restrictive than used, and that they gives values of bits in addition to the differences.
3. Our attack gives a set of sufficient conditions which ensure the differential to occur.
4. Our attack use a message modification technique to greatly improve the collision probability.

3.2 Differential Attacks on Hash Functions

The *difference* for two parameters X and X' is defined as $\Delta X = X' - X$. For any two messages M and M' with l-bit multiples, $M = (M_0, M_1, \cdots, M_{k-1})$, $M = (M_0{}', M_1{}', \cdots, M_{k-1}{}')$, a full *differential* for a hash function is defined as follows:

$$\Delta H_0 \overset{(M_0, M_0')}{\longrightarrow} \Delta H_1 \overset{(M_1, M_1')}{\longrightarrow} \Delta H_2 \overset{(M_2, M_2')}{\longrightarrow} \cdots\cdots \Delta H_{k-1} \overset{(M_{k-1}, M_{k-1}')}{\longrightarrow} \Delta H,$$

where ΔH_0 is the initial value difference which equals to zero. ΔH is the output difference for the two messages. $\Delta H_i = \Delta IV_i$ is the output difference for the i-th iteration, and also is the initial difference for the next iteration.

It is clear that if $\Delta H = 0$, there is a collision for M and M'. We call the differential that produces a collision *a collision differential*.

Provided that the hash function has 4 rounds, and each round has 16 step operations. For more details, we can represent the i-th iteration differential $\Delta H_i \overset{(M_i, M_i')}{\longrightarrow} \Delta H_{i+1}$ as follows:

$$\Delta H_i \xrightarrow{P_1} \Delta R_{i+1,1} \xrightarrow{P_2} \Delta R_{i+1,2} \xrightarrow{P_3} \Delta R_{i+1,3} \xrightarrow{P_4} \Delta R_{i+1,4} = \Delta H_{i+1}.$$

The round differential $\Delta R_{j-1} \longrightarrow \Delta R_j (j = 1, 2, 3, 4)$ with the probability P_j is expanded to the following differential characteristics.

$$\Delta R_{j-1} \xrightarrow{P_{j1}} \Delta X_1 \xrightarrow{P_{j2}} \cdots\cdots \xrightarrow{P_{j16}} \Delta X_{16} = \Delta R_j,$$

where $\Delta X_{t-1} \xrightarrow{P_{jt}} \Delta X_t, t = 1, 2, \cdots\cdots, 16$ is the differential characteristic in the t-th step of j-th round.

The probability P of the differential $\Delta H_i \overset{(M_i, M_i')}{\longrightarrow} \Delta H_{i+1}$ satisfies

$$P \geq \prod_{i=1}^{4} P_j \text{ and } P_j \geq \prod_{t=1}^{16} P_{jt}.$$

3.3 Optimized Collision Differentials for Hash Functions

In Section 3.1, we mentioned that our attack uses a message modification technique to improve the collision probability. According to the modification technique, we can get a rough method to search for optimized differentials (including collision differentials) of a hash function.

There are two kinds of message modifications:

1. For any two message blocks (M_i, M_i') and a 1-st round non-zero differential

$$\Delta H_i \overset{(M_i, M_i')}{\longrightarrow} \Delta R_{i+1,1}.$$

   Our attack can easily modify M_i to guarantee the 1-st round differential to hold with probability $P_1 = 1$.
2. Using multi-message modification techniques, we can not only guarantee the first-round differential to hold with the probability 1, but also improve the second-round differential probability greatly.

To find an optimized differential for a hash function, it is better to select a message block difference which results in a last two-round differential with a high probability.

4 Differential Attack on MD5

4.1 Notation

Before presenting our attack, we first introduce some notation to simplify the discussion.

1. $M = (m_0, m_1, ..., m_{15})$ and $M' = (m_0', m_1', ..., m_{15}')$ represent two 512-bit messages. $\Delta M = (\Delta m_0, \Delta m_1, ..., \Delta m_{15})$ denotes the difference of two message blocks. That is, $\Delta m_i = m_i' - m_i$ is the $i-th$ word difference.
2. a_i, d_i, c_i, b_i respectively denote the outputs of the $(4i-3)$-th, $(4i-2)$-th $(4i-1)$-th and $4i$-th steps for compressing M, where $1 \leq i \leq 16$. a_i', b_i', c_i', d_i' are defined similarly.
3. $a_{i,j}$, $b_{i,j}$, $c_{i,j}$, $d_{i,j}$ represent respectively the $j-th$ bit of a_i, b_i, c_i, d_i, where the least significant bit is the 1-st bit, and the most significant bit is 32-th bit.
4. $\phi_{i,j}$ is the j-th bit of the output for the nonlinear function ϕ_i in the i-th step operation.
5. $\Delta x_{i,j} = x'_{i,j} - x_{i,j} = \pm 1$ is the bit difference that is produced by changing the j-bit of x_i. $x_i[j]$, $x_i[-j]$ (x can be a, b, c, d, ϕ) is the resulting values by only changing the $j-th$ bit of the word x_i. $x_i[j]$ is obtained by changing the j-th bit of x_i from 0 to 1, and $x_i[-j]$ is obtained by changing the j-th bit of x_i from 1 to 0.
6. $\Delta x_i[j_1, j_2, ..., j_l] = x_i[j_1, j_2, ..., j_l] - x_i$ denotes the difference that is produced by the changes of $j_1 - th$, $j_2 - th$, ..., $j_l - th$ bits of x_i. $x_i[\pm j_1, \pm j_2, ..., \pm j_l]$ is the value by change $j_1 - th$, $j_2 - th$, ..., $j_l - th$ bits of x_i. The "+" sign (usually is omitted) means that the bit is changed from 0 to 1, and the "–" sign means that the bit is changed from 1 to 0.

4.2 Collision Differentials for MD5

Our attack can find many real collisions which are composed of two 1024-bit messages (M_0, M_1) and $(M_0{}', M_1{}')$) with the original initial value IV_0 of MD5:

$IV_0 :\ a_0 = \texttt{0x67452301},\ b_0 = \texttt{0xefcdab89},\ c_0 = \texttt{0x98badcfe},\ d_0 = \texttt{0x10325476}.$

We select a collision differential with two iterations as follows:

$$\Delta H_0 \overset{(M_0, M_0')}{\longrightarrow} \Delta H_1 \overset{(M_1, M_1')}{\longrightarrow} \Delta H = 0$$

where

$$\Delta M_0 = M_0' - M_0 = (0,0,0,0,2^{31},0,0,0,0,0,0,2^{15},0,0,2^{31},0)$$
$$\Delta M_1 = M_1' - M_1 = (0,0,0,0,2^{31},0,0,0,0,0,0,-2^{15},0,0,2^{31},0)$$
$$\Delta H_1 = (2^{31}, 2^{31}+2^{25}, 2^{31}+2^{25}, 2^{31}+2^{25}).$$

Non-zero entries of ΔM_0 and ΔM_1 are located at positions 5, 12 and 15. $\Delta H_1 = (\Delta a, \Delta b, \Delta c, \Delta d)$ is the difference of the four chaining values (a, d, c, b) after the first iteration.

We select ΔM_0 to ensure that both 3-4 round differential happens with a high probability. ΔM_1 is selected not only to ensure both 3-4 round differential happens with a high probability, but also to produce an output difference that can be cancelled with the output difference ΔH_1.

The collision differential with all the characteristics can be referred to Table 3 and Table 5. The columns of both tables have the same meanings. We just give the explanation for Table 3. The first column denotes the step, the second column is the chaining variable in each step for M_0, the third is the message word for M_0 in each step, the fourth is shift rotation, the fifth and the sixth are respectively the message word difference and chaining variable difference for M_0 and M_0', and the seventh is the chaining variable for M_0'. Especially, the empty items both in sixth and fifth columns denote zero differences, and steps those aren't listed in the table have zero differences both for message words and chaining variables.

4.3 Sufficient Conditions for the Characteristics to Hold

In what follows, we describe how to derive a set of sufficient conditions that guarantee the differential characteristic in Step 8 of MD5 (Table 3) to hold. Other conditions can be derived similarly.

The differential characteristic in Step 8 of MD5 is:

$$(\Delta c_2, \Delta d_2, \Delta a_2, \Delta b_1) \longrightarrow \Delta b_2.$$

Each chaining variable satisfies one of the following equations.

$b_1' = b_1$
$a_2' = a_2[7, ..., 22, -23]$
$d_2' = d_2[-7, 24, 32]$
$c_2' = c_2[7, 8, 9, 10, 11, -12, -24, -25, -26, 27, 28, 29, 30, 31, 32, 1, 2, 3, 4, 5, -6]$
$b_2' = b_2[1, 16, -17, 18, 19, 20, -21, -24]$

According to the operations in the 8-th step, we have

$$b_2 = c_2 + ((b_1 + F(c_2, d_2, a_2) + m_7 + t_7) \lll 22$$

$$b_2' = c_2' + ((b_1 + F(c_2', d_2', a_2') + m_7' + t_7) \lll 22$$

$$\phi_7 = F(c_2, d_2, a_2) = (c_2 \wedge d_2) \vee (\neg c_2 \wedge a_2)$$

In the above operations, c_2 occurs twice in the right hand side of the equation. In order to distinguish the two, let c_2^F denote the c_2 inside F, and c_2^{NF} denote the c_2 outside F.

The derivation is based on the following two facts:

1. Since $\Delta b_1 = 0$ and $\Delta m_7 = 0$, we know that $\Delta b_2 = \Delta c_2^{NF} + (\Delta\phi_7 \lll 22)$.
2. Fix one or two of the variables in F so that F is reduced to a single variable.

We get a set of sufficient conditions that ensure the differential characteristic holds.

1. **The conditions for each of the non-zero bits in Δb_2.**
   (a) The conditions $d_{2,11} = 1$ and $b_{2,1} = 0$ ensure the change of 1-st bit of b_2.
      i. If $d_{2,11} = \overline{a_{2,11}} = 1$, we know that $\Delta\phi_{7,11} = 1$.
      ii. After $\lll 22$, $\Delta\phi_{7,11}$ is in the position 1.
      iii. Since $\Delta c_{2,1}^{NF} = 0$, so, $\Delta b_{2,1} = \Delta c_{2,1}^{NF} + \Delta\phi_{7,11} = 1$.
   (b) The conditions $d_{2,26} = \overline{a_{2,26}} = 1$, $b_{2,16} = 0$ and $b_{2,17} = 1$ ensure the changes of 16-th bit and 17-th bit of b_2.
   (c) The conditions $d_{2,28} = \overline{a_{2,28}} = 0$, $b_{2,i} = 0, i = 18, 19, 20$ and $b_{2,21} = 1$ ensure the changes of 18-th, 19-th, 20-th, 21-th bits of b_2.
   (d) The conditions $d_{2,3} = \overline{a_{2,3}} = 0$ and $b_{2,24} = 1$ ensure the change of 24-th bit of b_2. This can be proven by the equation:

$$\Delta c_2^{NF}[-24, -25, -26, 27] + (\Delta\phi_7[3] \lll 22) = 2^{23} - 2^{24} = -2^{23}.$$

2. **The conditions for each of the zero bits in Δb_2.**
   (a) The condition $c_{2,17} = 0$ ensures the changed bits from 7-th bit to 12-th bit in $c_2'^{NF}$ and 17-th bit of a_2' result in no bit change in b_2. It is easily proven by the following equation:

$$\Delta c_2^{NF}[7, \ldots 11, -12] + (\Delta\phi_7[17] \lll 22) = -2^6 + 2^6 = 0.$$

   (b) The conditions $d_{2,i} = a_{2,i}$ ensure that the changed i-th bit in c_2^F result in no change in b_2, where $i \in \{1, 2, 4, 5, 25, 27, 29, 30, 31\}$.
   (c) The conditions $c_{2,i} = 1$ ensure that the changed i-th bit in a_2 result in no change in b_2, where $i \in \{13, 14, 15, 16, 18, 19, 20, 21, 22, 23\}$.
   (d) The condition $d_{2,6} = \overline{a_{2,6}} = 0$ ensures that the 6-th bit in c_2^F result in no change in b_2.
   (e) The condition $a_{2,32} = 1$ ensures that the changed 32-th bit in c_2^F and the 32-th bit in d_2 result in no change in b_2.
   (f) The condition $d_{2,i} = 0$ ensures that the changed i-th bit in a_2 and the i-th bit in c_2^F result in no change in b_2, where $i \in \{8, 9, 10\}$.

(g) The condition $d_{2,12} = 1$ ensures that the changed 12-th bit in a_2 and the 12-th bit in c_2^F result in no change in b_2.
(h) The condition $a_{2,24} = 0$ ensures that the changed 24-th bit in c_2^F and the 24-th bit in d_2 result in no change in b_2.
(i) The changed 7-th bits in c_2^F, d_2 and a_2 result in no change in b_2.

By the similar method, we can derive a set of sufficient conditions (see Table 4 and Table 6) which guarantee all the differential characteristics in the collision differential to hold.

4.4 Message Modification

Single-Message Modification. In order to make the attack efficient, it is very attractive to improve over the probabilistic method that we describe, by fixing some of the message words to a prior fulfilling some of the conditions. We observe that it is very easy to generate messages that fulfill all the conditions of the first 16 steps of MD5. We call it *single-message modification*.

For each message block M_0 (or similarly M_1) and intermediate values (H_0, or for the second block H_1 and H_1'), we apply the following procedures to modify M_0 (or M_1, respectively), so that all the conditions of round 1 (the first 16 steps) in Table 4 and Table 6 hold.

It is easy to modify M_0 such that the conditions of round 1 in Table 4 hold with probability 1.

For example, to ensure that 3 conditions for c_1 in Table 4 hold, we modify m_2 as follows:

$$c_1^{new} \leftarrow c_1^{old} - c_{1,7}^{old} \cdot 2^6 - c_{1,12}^{old} \cdot 2^{11} - c_{1,20}^{old} \cdot 2^{19}$$

$$m_2^{new} \leftarrow ((c_1^{new} - c_1^{old}) \ggg 17) + m_2^{old}.$$

By modifying each message word of message M_0, all the conditions in round 1 of Table 4 hold. The first iteration differential hold with probability 2^{-43}.

The same modification is applied to M_1. After modification, the second iteration differential hold with probability 2^{-37}.

Multi-message Modification. We further observe that it is even possible to fulfill a part of the conditions of the first 32 steps by an *multi-message modification*.

For example, if $a_{5,32} = 1$, we correct it into $a_{5,32} = 0$ by modifying m_1, m_2,m_3, m_4,m_5 such that the modification generates a partial collision from 2-6 steps, and remains that all the conditions in round 1 hold. See Table 1. Some other conditions can be corrected by the similar modification technique or other more precise modification techniques. By our modification, 37 conditions in round 2-4 are undetermined in the table 4, and 30 conditions in round 2-4 are undetermined in the table 6. So, the 1-st iteration differential holds with probability 2^{-37}, and the second iteration differential holds with probability 2^{-30}.

Table 1. The Message Modification for Correcting $a_{5,32}$

			Modify m_i	a^{new}, b^{new}, c^{new}, d^{new}
2	m_1	12	$m_1 \longleftarrow m_1 + 2^{26}$	d_1^{new}, a_1, b_0, c_0
3	m_2	17	$m_2 \longleftarrow ((c_1 - d_1^{new}) \ggg 17) - c_0 - \phi_2(d_1^{new}, a_1, b_0) - t_2$	c_1, d_1^{new}, a_1, b_0
4	m_3	22	$m_3 \longleftarrow (b_1 - c_1) \ggg 22) - b_0 - \phi_3(c_1, d_1^{new}, a_1) - t_3$	b_1, c_1, d_1^{new}, a_1
5	m_4	7	$m_4 \longleftarrow ((a_2 - b_1) \ggg 7) - a_1 - \phi_4(b_1, c_1, d_1^{new} - t_4$	a_2, b_1, c_1, d_1^{new}
6	m_5	12	$m_5 \longleftarrow ((d_2 - a_2) \ggg 12) - d_1^{new} - \phi_5(a_2, b_1, c_1) - t_5$	d_2, a_2, b_1, c_1

4.5 The Differential Attack on MD5

From the above description, it is very easy to show our attack on MD5.

The following is to describe how to find a two-block collision, of the following form

$$H_0 \overset{(M_0, M_0'), 2^{-37}}{\longrightarrow} \Delta H_1 \overset{(M_1, M_1'), 2^{-30}}{\longrightarrow} \Delta H = 0.$$

1. Repeat the following steps until a first block is found
   (a) Select a random message M_0.
   (b) Modify M_0 by the message modification techniques described in the previous subsection.
   (c) Then, M_0 and $M_0' = M_0 + \Delta M_0$ produce the first iteration differential

   $$\Delta M_0 \longrightarrow (\Delta H_1, \Delta M_1)$$

   with the probability 2^{-37}.
   (d) Test if all the characteristics really hold by applying the compression function on M_0 and M_0'.
2. Repeat the following steps until a collision is found
   (a) Select a random message M_1.
   (b) Modify M_1 by the message modification techniques described in the previous subsection.
   (c) Then, M_1 and $M_1 + \Delta M_1$ generate the second iteration differential

   $$(\Delta H_1, \Delta M_1) \longrightarrow \Delta H = 0$$

   with the probability 2^{-30}.
   (d) Test if this pair of messages lead to a collision.

The complexity of finding (M_0, M_0') doesn't exceed the time of running 2^{39} MD5 operations. To select another message M_0 is only to change the last two words from the previous selected message M_0. So, finding (M_0, M_0') only needs about one-time single-message modification for the first 14 words. This time can be neglected. For each selected message M_0, it is only needs two-time single-message modifications for the last two words and 7-time multi-message modifications for correcting 7 conditions in the second round, and each multi-message modification only needs about a few step operations, so the total time for both kinds of modifications is not exceeds about two MD5 operations for each selected message.

According to the probability of the first iteration differential, it is easy to know that the complexity of finding (M_0, M_0') is not exceeds 2^{39} MD5 operations.

Similarly, we can show that the complexity of finding (M_1, M_1') is not exceeds 2^{32} MD5 operations.

Table 2. Two pairs of collision for MD5. H is the hash value with little-endian and no message padding, and H^* is the hash value with big-endian and message padding

M_0	2dd31d1	c4eee6c5	69a3d69	5cf9af98	87b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a417125	e8255108	9fc9cdf7	f2bd1dd9	5b3c3780
M_1	d11d0b96	9c7b41dc	f497d8e4	d555655a	c79a7335	cfdebf0	66f12930	8fb109d1
	797f2775	eb5cd530	baade822	5c15cc79	ddcb74ed	6dd3c55f	d80a9bb1	e3a7cc35
M_0'	2dd31d1	c4eee6c5	69a3d69	5cf9af98	7b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a41f125	e8255108	9fc9cdf7	72bd1dd9	5b3c3780
M_1'	d11d0b96	9c7b41dc	f497d8e4	d555655a	479a7335	cfdebf0	66f12930	8fb109d1
	797f2775	eb5cd530	baade822	5c154c79	ddcb74ed	6dd3c55f	580a9bb1	e3a7cc35
H	9603161f	a30f9dbf	9f65ffbc	f41fc7ef				
H^*	a4c0d35c	95a63a80	5915367d	cfe6b751				
M_0	2dd31d1	c4eee6c5	69a3d69	5cf9af98	87b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a417125	e8255108	9fc9cdf7	f2bd1dd9	5b3c3780
M_1	313e82d8	5b8f3456	d4ac6dae	c619c936	b4e253dd	fd03da87	6633902	a0cd48d2
	42339fe9	e87e570f	70b654ce	1e0da880	bc2198c6	9383a8b6	2b65f996	702af76f
M_0'	2dd31d1	c4eee6c5	69a3d69	5cf9af98	7b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a41f125	e8255108	9fc9cdf7	72bd1dd9	5b3c3780
M_1'	313e82d8	5b8f3456	d4ac6dae	c619c936	34e253dd	fd03da87	6633902	a0cd48d2
	42339fe9	e87e570f	70b654ce	1e0d2880	bc2198c6	9383a8b6	ab65f996	702af76f
H	8d5e7019	61804e08	715d6b58	6324c015				
H^*	79054025	255fb1a2	6e4bc422	aef54eb4				

Two collisions of MD5 are given in Table 2. It is noted that the two collisions start with the same 1-st 512-bit block, and that given a first block that satisfies all the required criteria, it is easy to find many second blocks M_1, M_1' which lead to collisions.

5 Summary

In this paper we described a powerful attack against hash functions, and in particular showed that finding a collision of MD5 is easily feasible.

Our attack is also able to break efficiently other hash functions, such as HAVAL-128, MD4, RIPEMD, and SHA-0. The analysis results for these hash functions are as follows:

1. The time complexity for finding a collision for MD4 is about 2^{23} MD4 operations without the multi-message modification, and is about 2^8 MD4 operations with the multi-message modification.

2. The time complexity for finding a collision for HAVAL-128 is about 2^{13} MD4 operations without the multi-message modification, and is 2^7 HAVAL-128 operations with the multi-message modification.
3. The time complexity for finding a collision for RIPEMD is about 2^{30} RIPEMD operations without the multi-message modification, and is 2^{18} RIPEMD operations with the multi-message modification.
4. The time complexity for finding a collision for SHA-0 is about 2^{61} SHA-0 operations without the multi-message modification, and is 2^{45} SHA-0 operations with the multi-message modification.

Acknowledgements

It is a pleasure to acknowledge Dengguo Feng for the conversations that led to this research on MD5. We would like to thank Eli Biham, Andrew C. Yao, and Yiqun Lisa Yin for their important advice, corrections, and suggestions, and for spending their precious time on our research. We would also like to thank Xuejia Lai, Hans Dobbertin, Magnus Daum for various discussions on this paper. The research is supported by the National Natural Science Foundation of China (Grant No. 90304009).

References

1. E. Biham, A. Shamir. Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.
2. E. Biham, R. Chen, *Near collision for SHA-0,* Advances in Cryptology, Crypto'04, 2004, LNCS 3152, pp. 290-305.
3. B. den. Boer, A. Bosselaers. Collisions for the compression function of MD5, Advances in Cryptology, Eurocrypt'93 Proceedings, Springer-Verlag, 1994.
4. F. Chabaud, A. Joux. Differential collisions in SHA-0, Advances in Cryptology, Crypto'98 Proceedings, Springer-Verlag, 1998.
5. S. Cotini, R.L. Rivest, M.J.B. Robshaw, Y. Lisa Yin. Security of the RC6^{TM} Block Cipher, `http://www.rsasecurity.com/rsalabs/rc6/`.
6. I. B. Damgard. A design principle for hash functions, Advances in Cryptology, Crypto'89 Proceedings, Springer-Verlag, 1990.
7. H. Dobbertin. Cryptanalysis of MD4, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996, 53-69.
8. H. Dobbertin. Cryptanalysis of MD5 compress, presented at the rump session of Eurocrypt'96.
9. H. Dobbertin. The status of MD5 after a recent attack, CryptoBytes 2 (2), 1996, `ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf`.
10. H. Dobbertin. RIPEMD with two round compress function is not collision-free, Journal of Cryptology, 10:51-69, 1997.
11. H. Dobbertin, A. Bosselaers, B. Preneel. RIPEMD-160: A strengthened version of RIPEMD, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996.
12. FIPS 180-1. Secure hash standard, NIST, US Department of Commerce, Washington D.C., Springer-Verlag, 1996.

13. FIPS 180-2. Secure Hash Standard, `http://csrc.nist.gov/publications/`, 2002.
14. A. Joux. Collisions for SHA-0, rump session of Crypto'04, 2004.
15. RIPE. Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040), LNCS 1007, Springer-Verlag, 1995.
16. R.C. Merkle. One way hash function and DES, Advances in Cryptology, Crypto'89 Proceedings, Springer-Verlag, 1990.
17. R.L. Rivest. The MD4 message digest algorithm, Advances in Cryptology, Crypto'90, Springer-Verlag, 1991, 303-311.
18. R.L. Rivest. The MD5 message-digest algorithm, Request for Comments (RFC 1320), Internet Activities Board, Internet Privacy Task Force, 1992.
19. X.Y. Wang, F.D. Guo, X.J. Lai, H.B. Yu, Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD, rump session of Crypto'04, E-print, 2004.
20. Y.L. Zheng, J. Pieprzyk, J. Seberry. HAVAL–A one-way hashing algorithm with variable length of output, Advances in Cryptology, Auscrypt'92 Proceedings, Springer-Verlag.

Table 3. The Differential Characteristics in the First Iteration Differential

Step	The output in i-th step for M_0	w_i	s_i	Δw_i	The output difference in i-th step	The output in i-th step for M_0'
4	b_1	m_3	22			
5	a_2	m_4	7	2^{31}	-2^6	$a_2[7,\ldots,22,-23]$
6	d_2	m_5	12		$-2^6+2^{23}+2^{31}$	$d_2[-7,24,32]$
7	c_2	m_6	17		$-1-2^6+2^{23}-2^{27}$	$c_2[7,8,9,10,11,-12,-24,-25,-26,27,28,29,30,31,32,1,2,3,4,5,-6]$
8	b_2	m_7	22		$1-2^{15}-2^{17}-2^{23}$	$b_2[1,16,-17,18,19,20,-21,-24]$,
9	a_3	m_8	7		$1-2^6+2^{31}$	$a_3[-1,2,7,8,-9,-32]$
10	d_3	m_9	12		$2^{12}+2^{31}$	$d_3[-13,14,32]$
11	c_3	m_{10}	17		$2^{30}+2^{31}$	$c_3[31,32]$
12	b_3	m_{11}	22	2^{15}	$-2^7-2^{13}+2^{31}$	$b_3[8,-9,14,\ldots,19,-20,32]$
13	a_4	m_{12}	7		$2^{24}+2^{31}$	$a_4[-25,26,32]$
14	d_4	m_{13}	12		2^{31}	$d_4[32]$
15	c_4	m_{14}	17	2^{31}	$2^3-2^{15}+2^{31}$	$c_4[4,-16,32]$
16	b_4	m_{15}	22		$-2^{29}+2^{31}$	$b_4[-30,32]$
17	a_5	m_1	5		2^{31}	$a_5[32]$
18	d_5	m_6	9		2^{31}	$d_5[32]$
19	c_5	m_{11}	14	2^{15}	$2^{17}+2^{31}$	$c_5[18,32]$
20	b_5	m_0	20		2^{31}	$b_5[32]$
21	a_6	m_5	5		2^{31}	$a_6[32]$
22	d_6	m_{10}	9		2^{31}	$d_6[32]$
23	c_6	m_{15}	14			c_6
24	b_6	m_4	20	2^{31}		b_6
25	a_7	m_9	5			a_7
26	d_7	m_{14}	9	2^{31}		d_7
27	c_7	m_3	14			c_7
...	...	...	...	...	...	...
34	d_9	m_8	11			d_9
35	c_9	m_{11}	16	2^{15}	2^{31}	$c_9[*32]$
36	b_9	m_{14}	23	2^{31}	2^{31}	$b_9[*32]$
37	a_{10}	m_1	4		2^{31}	$a_{10}[*32]$
38	d_{10}	m_4	11	2^{31}	2^{31}	$d_{10}[*32]$
39	c_{10}	m_7	16		2^{31}	$c_{10}[*32]$
...	...	...	...	...	...	...
45	a_{12}	m_9	4		2^{31}	$a_{12}[*32]$
46	d_{12}	m_{12}	11		2^{31}	$d_{12}[32]$
47	c_{12}	m_{15}	16		2^{31}	$c_{12}[32]$
48	b_{12}	m_2	23		2^{31}	$b_{12}[32]$
49	a_{13}	m_0	6		2^{31}	$a_{13}[32]$
50	d_{13}	m_7	10		2^{31}	$d_{13}[-32]$
51	c_{13}	m_{14}	15	2^{31}	2^{31}	$c_{13}[32]$
52	b_{13}	m_5	21		2^{31}	$b_{13}[-32]$
...	...	...	...	...	...	...
58	d_{15}	m_{15}	10		2^{31}	$d_{15}[-32]$
59	c_{15}	m_6	15		2^{31}	$c_{15}[32]$
60	b_{15}	m_{13}	21		2^{31}	$b_{15}[32]$
61	$aa_0=a_{16}+a_0$	m_4	6	2^{31}	2^{31}	$aa_0'=aa_0[32]$
62	$dd_0=d_{16}+d_0$	m_{11}	10	2^{15}	2^{31}	$dd_0'=dd_0[26,32]$
63	$cc_0=c_{16}+c_0$	m_2	15		2^{31}	$cc_0'=cc_0[-26,27,32]$
64	$bb_0=b_{16}+b_0$	m_9	21		2^{31}	$bb_0'=bb_0[26,-32]$

Table 4. A Set of Sufficient Conditions for the First Iteration Differential

c_1	$c_{1,7} = 0$, $c_{1,12} = 0$, $c_{1,20} = 0$
b_1	$b_{1,7} = 0$, $b_{1,8} = c_{1,8}$, $b_{1,9} = c_{1,9}$, $b_{1,10} = c_{1,10}$, $b_{1,11} = c_{1,11}$, $b_{1,12} = 1$, $b_{1,13} = c_{1,13}$, $b_{1,14} = c_{1,14}$, $b_{1,15} = c_{1,15}$, $b_{1,16} = c_{1,16}$, $b_{1,17} = c_{1,17}$, $b_{1,18} = c_{1,18}$,$b_{1,19} = c_{1,19}$, $b_{1,20} = 1$, $b_{1,21} = c_{1,21}$, $b_{1,22} = c_{1,22}$, $b_{1,23} = c_{1,23}$, $b_{1,24} = 0$, $b_{1,32} = 1$
a_2	$a_{2,1} = 1$, $a_{2,3} = 1$, $a_{2,6} = 1$, $a_{2,7} = 0$, $a_{2,8} = 0$, $a_{2,9} = 0$, $a_{2,10} = 0$, $a_{2,11} = 0$, $a_{2,12} = 0$, $a_{2,13} = 0$, $a_{2,14} = 0$, $a_{2,15} = 0$, $a_{2,16} = 0$, $a_{2,17} = 0$, $a_{2,18} = 0$, $a_{2,19} = 0$, $a_{2,20} = 0$, $a_{2,21} = 0$, $a_{2,22} = 0$, $a_{2,23} = 1$, $a_{2,24} = 0$, $a_{2,26} = 0$, $a_{2,28} = 1$, $a_{2,32} = 1$
d_2	$d_{2,1} = 1$, $d_{2,2} = a_{2,2}$, $d_{2,3} = 0$, $d_{2,4} = a_{2,4}$, $d_{2,5} = a_{2,5}$, $d_{2,6} = 0$, $d_{2,7} = 1$, $d_{2,8} = 0$, $d_{2,9} = 0$, $d_{2,10} = 0$, $d_{2,11} = 1$, $d_{2,12} = 1$, $d_{2,13} = 1$, $d_{2,14} = 1$, $d_{2,15} = 0$, $d_{2,16} = 1$, $d_{2,17} = 1$, $d_{2,18} = 1$, $d_{2,19} = 1$, $d_{2,20} = 1$, $d_{2,21} = 1$, $d_{2,22} = 1$, $d_{2,23} = 1$, $d_{2,24} = 0$, $d_{2,25} = a_{2,25}$, $d_{2,26} = 1$, $d_{2,27} = a_{2,27}$, $d_{2,28} = 0$, $d_{2,29} = a_{2,29}$, $d_{2,30} = a_{2,30}$, $d_{2,31} = a_{2,31}$, $d_{2,32} = 0$
c_2	$c_{2,1} = 0$, $c_{2,2} = 0$, $c_{2,3} = 0$, $c_{2,4} = 0$, $c_{2,5} = 0$, $c_{2,6} = 1$, $c_{2,7} = 0$, $c_{2,8} = 0$, $c_{2,9} = 0$, $c_{2,10} = 0$, $c_{2,11} = 0$, $c_{2,12} = 1$, $c_{2,13} = 1$, $c_{2,14} = 1$, $c_{2,15} = 1$, $c_{2,16} = 1$, $c_{2,17} = 0$, $c_{2,18} = 1$, $c_{2,19} = 1$, $c_{2,20} = 1$, $c_{2,21} = 1$, $c_{2,22} = 1$, $c_{2,23} = 1$, $c_{2,24} = 1$, $c_{2,25} = 1$, $c_{2,26} = 1$, $c_{2,27} = 0$, $c_{2,28} = 0$, $c_{2,29} = 0$, $c_{2,30} = 0$, $c_{2,31} = 0$, $c_{2,32} = 0$
b_2	$b_{2,1} = 0$, $b_{2,2} = 0$, $b_{2,3} = 0$, $b_{2,4} = 0$, $b_{2,5} = 0$, $b_{2,6} = 0$, $b_{2,7} = 1$, $b_{2,8} = 0$, $b_{2,9} = 1$, $b_{2,10} = 0$, $b_{2,11} = 1$, $b_{2,12} = 0$, $b_{2,14} = 0$, $b_{2,16} = 0$, $b_{2,17} = 1$, $b_{2,18} = 0$, $b_{2,19} = 0$, $b_{2,20} = 0$, $b_{2,21} = 1$, $b_{2,24} = 1$, $b_{2,25} = 1$, $b_{2,26} = 0$, $b_{2,27} = 0$, $b_{2,28} = 0$, $b_{2,29} = 0$, $b_{2,30} = 0$, $b_{2,31} = 0$, $b_{2,32} = 0$
a_3	$a_{3,1} = 1$, $a_{3,2} = 0$, $a_{3,3} = 1$, $a_{3,4} = 1$, $a_{3,5} = 1$, $a_{3,6} = 1$, $a_{3,7} = 0$, $a_{3,8} = 0$, $a_{3,9} = 1$, $a_{3,10} = 1$, $a_{3,11} = 1$, $a_{3,12} = 1$, $a_{3,13} = b_{2,13}$, $a_{3,14} = 1$, $a_{3,16} = 0$, $a_{3,17} = 0$, $a_{3,18} = 0$, $a_{3,19} = 0$, $a_{3,20} = 0$, $a_{3,21} = 1$, $a_{3,25} = 1$, $a_{3,26} = 1$, $a_{3,27} = 0$, $a_{3,28} = 1$, $a_{3,29} = 1$, $a_{3,30} = 1$, $a_{3,31} = 1$, $a_{3,32} = 1$
d_3	$d_{3,1} = 0$, $d_{3,2} = 0$, $d_{3,7} = 1$, $d_{3,8} = 0$, $d_{3,9} = 0$, $d_{3,13} = 1$, $d_{3,14} = 0$, $d_{3,16} = 1$, $d_{3,17} = 1$, $d_{3,18} = 1$, $d_{3,19} = 1$, $d_{3,20} = 1$, $d_{3,21} = 1$, $d_{3,24} = 0$, $d_{3,31} = 1$, $d_{3,32} = 0$
c_3	$c_{3,1} = 0$, $c_{3,2} = 1$, $c_{3,7} = 1$, $c_{3,8} = 1$, $c_{3,9} = 0$, $c_{3,13} = 0$, $c_{3,14} = 0$, $c_{3,15} = d_{3,15}$, $c_{3,17} = 1$, $c_{3,18} = 0$, $c_{3,19} = 0$, $c_{3,20} = 0$, $c_{3,16} = 1$, $c_{3,31} = 0$, $c_{3,32} = 0$
b_3	$b_{3,8} = 0$, $b_{3,9} = 1$, $b_{3,13} = 1$, $b_{3,14} = 0$, $b_{3,15} = 0$, $b_{3,16} = 0$, $b_{3,17} = 0$, $b_{3,18} = 0$, $b_{3,20} = 1$, $b_{3,25} = c_{3,25}$, $b_{3,26} = c_{3,26}$, $b_{3,19} = 0$, $b_{3,31} = 0$, $b_{3,32} = 0$
a_4	$a_{4,4} = 1$, $a_{4,8} = 0$, $a_{4,9} = 0$, $a_{4,14} = 1$, $a_{4,15} = 1$, $a_{4,16} = 1$, $a_{4,17} = 1$, $a_{4,18} = 1$, $a_{4,20} = 1$, $a_{4,25} = 1$, $a_{4,26} = 0$, $a_{4,31} = 1$, $a_{4,19} = 1$, $a_{4,32} = 0$
d_4	$d_{4,4} = 1$, $d_{4,8} = 1$, $d_{4,9} = 1$, $d_{4,14} = 1$, $d_{4,15} = 1$, $d_{4,16} = 1$, $d_{4,17} = 1$, $d_{4,18} = 1$, $d_{4,19} = 0$, $d_{4,20} = 1$, $d_{4,25} = 0$, $d_{4,26} = 0$, $d_{4,30} = 0$, $d_{4,32} = 0$
c_4	$c_{4,4} = 0$, $c_{4,16} = 1$, $c_{4,25} = 1$, $c_{4,26} = 0$, $c_{4,30} = 1$, $c_{4,32} = 0$
b_4	$b_{4,30} = 1$, $b_{4,32} = 0$
a_5	$a_{5,4} = b_{4,4}$, $a_{5,16} = b_{4,16}$, $a_{5,18} = 0$, $a_{5,32} = 0$
d_5	$d_{5,18} = 1$, $d_{5,30} = a_{5,30}$, $d_{5,32} = 0$
c_5	$c_{5,18} = 0$, $c_{5,32} = 0$
b_5	$b_{5,32} = 0$
$a_6 - b_6$	$a_{6,18} = b_{5,18}$, $a_{6,32} = 0$, $d_{6,32} = 0$, $c_{6,32} = 0$, $b_{6,32} = c_{6,32} + 1$
c_9, b_{12}	$\phi_{34,32} = 0$, $b_{12,32} = d_{12,32}$
$a_{13} - b_{13}$	$a_{13,32} = c_{12,32}$, $d_{13,32} = b_{12,32} + 1$, $c_{13,32} = a_{13,32}$, $b_{13,32} = d_{13,32}$
$a_{14} - b_{14}$	$a_{14,32} = c_{13,32}$, $d_{14,32} = b_{13,32}$, $c_{14,32} = a_{14,32}$, $b_{14,32} = d_{14,32}$
a_{15}	$a_{15,32} = c_{14,32}$
d_{15}	$d_{15,32} = b_{14,32}$
c_{15}	$c_{15,32} = a_{15,32}$
b_{15}	$b_{15,26} = 0$, $b_{15,32} = d_{15,32} + 1$
$aa_0 = a_{16} + a_0$	$a_{16,26} = 1$, $a_{16,27} = 0$, $a_{16,32} = c_{15,32}$
$dd_0 = d_{16} + d_0$	$dd_{0,26} = 0$, $d_{16,32} = b_{15,32}$
$cc_0 = c_{16} + c_0$	$cc_{0,26} = 1$, $cc_{0,27} = 0$, $cc_{0,32} = dd_{0,32}$, $c_{16,32} = d_{16,32}$
$bb_0 = b_{16} + b_0$	$bb_{0,26} = 0, bb_{0,27} = 0, bb_{0,6} = 0, bb_{0,32} = cc_{0,32}$

Table 5. All the Differential Characteristics in the Second Iteration Differential

Step	The output in i-th step for M_1	w_i	s_i	Δw_i	The output Difference in i-th step	The output in i-th step for M_1'
IV	aa_0, dd_0 cc_0, bb_0					$aa_0[32]$, $dd_0[26,32]$ $cc_0[-26,27,32]$, $bb_0[26,-32]$
1	a_1	m_0	7		$2^{25}+2^{31}$	$a_1[26,-32]$
2	d_1	m_1	12		$2^5+2^{25}+2^{31}$	$d_1[6,26,-32]$
3	c_1	m_2	17		$2^5+2^{11}+2^{16}$ $+2^{25}+2^{31}$	$c_1[-6,-7,8,-12,13,$ -17,. . . ,-21,22,-26,. . . ,-30,31,-32]
4	b_1	m_3	22		$-2+2^5+2^{25}+2^{31}$	$b_1[2,3,4,-5,6,-26,27,-32]$
5	a_2	m_4	7	2^{31}	$1+2^6+2^8+2^9+2^{31}$	$a_2[1,-7,8,9,-10,-11,-12,13,32]$
6	d_2	m_5	12		$-2^{16}-2^{20}+2^{31}$	$d_2[17,-18,21,-22,32]$
7	c_2	m_6	17		$-2^6-2^{27}+2^{31}$	$c_2[7,8,9,-10,28,-29,-32]$
8	b_2	m_7	22		$2^{15}-2^{17}-2^{23}+2^{31}$	$b_2[-16,17,-18,24,25,26,-27,-32]$
9	a_3	m_8	7		$1+2^6+2^{31}$	$a_3[-1,2,-7,-8,-9,10,-32]$
10	d_3	m_9	12		$2^{12}+2^{31}$	$d_3[13,-32]$
11	c_3	m_{10}	17		2^{31}	$c_3[-32]$
12	b_3	m_{11}	22	-2^{15}	$-2^7-2^{13}+2^{31}$	$b_3[-8,14,15,16,17,18,19,-20,-32]$
13	a_4	m_{12}	7		$2^{24}+2^{31}$	$a_4[-25,\ldots,-30,31,32]$
14	d_4	m_{13}	12		2^{31}	$d_4[32]$
15	c_4	m_{14}	17	2^{31}	$2^3+2^{15}+2^{31}$	$c_4[4,16,32]$
16	b_4	m_{15}	22		$-2^{29}+2^{31}$	$b_4[-30,32]$
17	a_5	m_1	5		2^{31}	$a_5[32]$
18	d_5	m_6	9		2^{31}	$d_5[32]$
19	c_5	m_{11}	14	-2^{15}	$2^{17}+2^{31}$	$c_5[18,32]$
20	b_5	m_0	20		2^{31}	$b_5[32]$
21	a_6	m_5	5		2^{31}	$a_6[32]$
22	d_6	m_{10}	9		2^{31}	$d_6[32]$
23	c_6	m_{15}	14			$c_6[32]$
24	b_6	m_4	20	2^{31}		$b_6[32]$
25	a_7	m_9	5			a_7
26	d_7	m_{14}	9	2^{31}		d_7
27	c_7	m_3	14			c_7
. . .	. . .	. . .	. . .	. . .	. . .	. . .
34	d_9	m_8	11			d_9
35	c_9	m_{11}	16	-2^{15}	2^{31}	$c_9[*32]$
36	b_9	m_{14}	23	2^{31}	2^{31}	$d_9[*32]$
37	a_{10}	m_1	4		2^{31}	$a_{10}[*32]$
38	d_{10}	m_4	11	2^{31}	2^{31}	$d_{10}[*32]$
39	c_{10}	m_7	16		2^{31}	$c_{10}[*32]$
. . .	. . .	. . .	. . .	. . .	. . .	. . .
49	a_{13}	m_0	6		2^{31}	$a_{13}[32]$
50	d_{13}	m_7	10		2^{31}	$d_{13}[-32]$
51	c_{13}	m_{14}	15	2^{31}	2^{31}	$c_{13}[32]$
52	b_{13}	m_5	21		2^{31}	$b_{13}[-32]$
. . .	. . .	. . .	. . .	. . .	. . .	. . .
59	c_{15}	m_6	15		2^{31}	$c_{15}[32]$
60	b_{15}	m_{13}	21		2^{31}	$b_{15}[32]$
61	$a_{16}+aa_0$	m_4	6	2^{31}		$a_{16}+aa_0=a'_{16}+aa'_0$
62	$d_{16}+dd_0$	m_{11}	10	-2^{15}		$d_{16}+dd_0=d'_{16}+dd'_0$
63	$c_{16}+cc_0$	m_2	15			$c_{16}+cc_0=c'_{16}+cc'_0$
64	$b_{16}+bb_0$	m_9	21			$b_{16}+bb_0=b'_{16}+bb'_0$

Table 6. A Set of Sufficient Conditions for the Second Iteration Differential

a_1	$a_{1,6}=0$, $a_{1,12}=0$, $a_{1,22}=1$, $a_{1,26}=0$, $a_{1,27}=1$, $a_{1,28}=0$, $a_{1,32}=1$
d_1	$d_{1,2}=0$, $d_{1,3}=0$, $d_{1,6}=0$, $d_{1,7}=a_{1,7}$, $d_{1,8}=a_{1,8}$, $d_{1,12}=1$, $d_{1,13}=a_{1,13}$, $d_{1,16}=0$, $d_{1,17}=a_{1,17}$, $d_{1,18}=a_{1,18}$, $d_{1,19}=a_{1,19}$, $d_{1,20}=a_{1,20}$, $d_{1,21}=a_{1,21}$, $d_{1,22}=0$, $d_{1,26}=0$, $d_{1,27}=1$, $d_{1,28}=1$, $d_{1,29}=a_{1,29}$, $d_{1,30}=a_{1,30}$, $d_{1,31}=a_{1,31}$, $d_{1,32}=1$
c_1	$c_{1,2}=1$, $c_{1,3}=1$, $c_{1,4}=d_{1,4}$, $c_{1,5}=d_{1,5}$, $c_{1,6}=1$, $c_{1,7}=1$, $c_{1,8}=0$, $c_{1,9}=1$, $c_{1,12}=1$, $c_{1,13}=0$, $c_{1,17}=1$, $c_{1,18}=1$, $c_{1,19}=1$, $c_{1,20}=1$, $c_{1,21}=1$, $c_{1,22}=0$, $c_{1,26}=1$, $c_{1,27}=1$, $c_{1,28}=1$, $c_{1,29}=1$, $c_{1,30}=1$, $c_{1,31}=0$, $c_{1,32}=1$
b_1	$b_{1,1}=c_{1,1}$, $b_{1,2}=0$, $b_{1,3}=0$, $b_{1,4}=0$, $b_{1,5}=1$, $b_{1,6}=0$, $b_{1,7}=0$, $b_{1,8}=0$, $b_{1,9}=0$, $b_{1,10}=c_{1,10}$, $b_{1,11}=c_{1,11}$, $b_{1,12}=0$, $b_{1,13}=0$, $b_{1,17}=0$, $b_{1,18}=0$, $b_{1,19}=1$, $b_{1,20}=0$, $b_{1,21}=0$, $b_{1,22}=0$, $b_{1,26}=1$, $b_{1,27}=0$, $b_{1,28}=1$, $b_{1,29}=1$, $b_{1,30}=1$, $b_{1,31}=0$, $b_{1,32}=1$
a_2	$a_{2,1}=0$, $a_{2,2}=0$, $a_{2,3}=0$, $a_{2,4}=0$, $a_{2,5}=1$, $a_{2,6}=0$, $a_{2,7}=1$, $a_{2,8}=0$, $a_{2,9}=0$, $a_{2,10}=1$, $a_{2,11}=1$, $a_{2,12}=1$, $a_{2,13}=0$, $a_{2,17}=1$, $a_{2,18}=1$, $a_{2,19}=1$, $a_{2,20}=1$, $a_{2,27}=0$, $a_{2,28}=1$, $a_{2,29}=0$, $a_{2,30}=0$, $a_{2,21}=0$, $a_{2,22}=1$, $a_{2,31}=1$, $a_{2,32}=0$
d_2	$d_{2,1}=0$, $d_{2,2}=1$, $d_{2,3}=1$, $d_{2,4}=0$, $d_{2,5}=1$, $d_{2,6}=0$, $d_{2,7}=1$, $d_{2,8}=0$, $d_{2,9}=0$, $d_{2,10}=0$, $d_{2,11}=1$, $d_{2,12}=1$, $d_{2,13}=0$, $d_{2,17}=0$, $d_{2,18}=1$, $d_{2,21}=0$, $d_{2,22}=1$, $d_{2,26}=0$, $d_{2,27}=1$, $d_{2,28}=0$, $d_{2,29}=0$, $d_{2,32}=0$
c_2	$c_{2,1}=1$, $c_{2,7}=0$, $c_{2,8}=0$, $c_{2,9}=0$, $c_{2,10}=1$, $c_{2,11}=1$, $c_{2,12}=1$, $c_{2,13}=1$, $c_{2,16}=d_{2,16}$, $c_{2,17}=1$, $c_{2,18}=0$, $c_{2,21}=0$, $c_{2,22}=0$, $c_{2,24}=d_{2,24}$, $c_{2,25}=d_{2,25}$, $c_{2,26}=1$, $c_{2,27}=1$, $c_{2,28}=0$, $c_{2,29}=1$, $c_{2,32}=1$
b_2	$b_{2,1}=0$, $b_{2,2}=c_{2,2}$, $b_{2,7}=1$, $b_{2,8}=1$, $b_{2,9}=1$, $b_{2,10}=1$, $b_{2,16}=1$, $b_{2,17}=0$, $b_{2,18}=1$, $b_{2,21}=1$, $b_{2,22}=1$, $b_{2,24}=0$, $b_{2,25}=0$, $b_{2,26}=0$, $b_{2,27}=1$, $b_{2,28}=0$, $b_{2,29}=0$, $b_{2,32}=1$
a_3	$a_{3,1}=1$, $a_{3,2}=0$, $a_{3,7}=1$, $a_{3,8}=1$, $a_{3,9}=1$, $a_{3,10}=0$, $a_{3,13}=b_{2,13}$, $a_{3,16}=0$, $a_{3,17}=1$, $a_{3,18}=0$, $a_{3,24}=0$, $a_{3,25}=0$, $a_{3,26}=0$, $a_{3,27}=1$, $a_{3,28}=1$, $a_{3,29}=1$, $a_{3,32}=1$
d_3	$d_{3,1}=0$, $d_{3,2}=0$, $d_{3,7}=1$, $d_{3,8}=1$, $d_{3,9}=1$, $d_{3,10}=1$, $d_{3,13}=0$, $d_{3,16}=1$, $d_{3,17}=1$, $d_{3,18}=1$, $d_{3,19}=0$, $d_{3,24}=1$, $d_{3,25}=1$, $d_{3,26}=1$, $d_{3,27}=1$, $d_{3,32}=1$
c_3	$c_{3,1}=1$, $c_{3,2}=1$, $c_{3,7}=1$, $c_{3,8}=1$, $c_{3,9}=1$, $c_{3,10}=1$, $c_{3,13}=0$, $c_{3,14}=d_{3,14}$, $c_{3,15}=d_{3,15}$, $c_{3,16}=1$, $c_{3,17}=1$, $c_{3,18}=0$, $c_{3,19}=1$, $c_{3,20}=d_{3,20}$, $c_{3,32}=1$
b_3	$b_{3,8}=1$, $b_{3,13}=1$, $b_{3,14}=0$, $b_{3,15}=0$, $b_{3,16}=0$, $b_{3,17}=0$, $b_{3,18}=0$, $b_{3,19}=0$, $b_{3,20}=1$, $b_{3,25}=c_{3,25}$, $b_{3,26}=c_{3,26}$, $b_{3,27}=c_{3,27}$, $b_{3,28}=c_{3,28}$, $b_{3,29}=c_{3,29}$, $b_{3,30}=c_{3,30}$, $b_{3,31}=c_{3,31}$, $b_{3,32}=1$
a_4	$a_{4,4}=1$, $a_{4,8}=0$, $a_{4,14}=1$, $a_{4,15}=1$, $a_{4,16}=1$, $a_{4,17}=1$, $a_{4,18}=1$, $a_{4,19}=1$, $a_{4,20}=1$ $a_{4,25}=1$, $a_{4,26}=1$, $a_{4,27}=1$, $a_{4,28}=1$, $a_{4,29}=1$, $a_{4,30}=1$, $a_{4,31}=0$, $a_{4,32}=0$
d_4	$d_{4,4}=1$, $d_{4,8}=1$, $d_{4,14}=1$, $d_{4,15}=1$, $d_{4,16}=1$, $d_{4,17}=1$, $d_{4,18}=1$, $d_{4,19}=0$, $d_{4,20}=1$ $d_{4,25}=0$, $d_{4,26}=0$, $d_{4,27}=0$, $d_{4,28}=0$, $d_{4,29}=0$, $d_{4,30}=0$, $d_{4,31}=1$, $d_{4,32}=0$
c_4	$c_{4,4}=0$, $c_{4,16}=0$, $c_{4,25}=1$, $c_{4,26}=0$, $c_{4,27}=1$, $c_{4,28}=1$, $c_{4,29}=1$, $c_{4,30}=1$ $c_{4,31}=1$, $c_{4,32}=0$
b_4	$b_{4,30}=1$, $b_{4,32}=0$
a_5	$a_{5,4}=b_{4,4}$, $a_{5,16}=b_{4,16}$, $a_{5,18}=0$, $a_{5,32}=0$
d_5	$d_{5,18}=1$, $d_{5,30}=a_{5,30}$, $d_{5,32}=0$
c_5	$c_{5,18}=0$, $c_{5,32}=0$
b_5	$b_{5,32}=0$,
a_6-b_6	$a_{6,18}=b_{5,18}$, $a_{6,32}=0$, $d_{6,32}=0$, $c_{6,32}=0$, $b_{6,32}=c_{6,32}+1$
c_9, b_{12}	$\phi_{34,32}=1$, $b_{12,32}=d_{12,32}$,
$a_{13}-b_{13}$	$a_{13,32}=c_{12,32}$, $d_{13,32}=b_{12,32}+1$, $c_{13,32}=a_{13,32}$, $b_{13,32}=d_{13,32}$
$a_{14}-b_{14}$	$a_{14,32}=c_{13,32}$, $d_{14,32}=b_{13,32}$, $c_{14,32}=a_{14,32}$, $b_{14,32}=d_{14,32}$
$a_{15}-b_{15}$	$a_{15,32}=c_{14,32}$, $d_{15,32}=b_{14,32}$, $c_{15,32}=a_{15,32}$, $b_{15,32}=d_{15,32}+1$
a_{16}	$a_{16,26}=1$, $a_{16,32}=c_{15,32}$
d_{16}	$d_{16,26}=1$,$d_{16,32}=b_{15,32}$
c_{16}	$c_{16,26}=1$,$c_{16,32}=a_{16,32}$
b_{16}	$b_{16,26}=1$

Collisions of SHA-0 and Reduced SHA-1*

Eli Biham[1,**], Rafi Chen[1], Antoine Joux[2,3,***],
Patrick Carribault[3], Christophe Lemuet[3], and William Jalby[3]

[1] Computer Science Department,
Technion – Israel Institute of Technology,
Haifa 32000, Israel
{biham, rafi_hen}@cs.technion.ac.il
http://www.cs.technion.ac.il/~biham/
[2] DGA
antoine.joux@m4x.org
[3] Laboratoire PRISM[†],
Université de Versailles St-Quentin-en-Yvelines,
45, avenue des Etats-Unis,
78035 Versailles Cedex, France
{Patrick.Carribault, Christophe.Lemuet, William.Jalby}@prism.uvsq.fr

Abstract. In this paper we describe improvements to the techniques used to cryptanalyze SHA-0 and introduce the first results on SHA-1. The results include a generic multi-block technique that uses near-collisions in order to find collisions, and a four-block collision of SHA-0 found using this technique with complexity 2^{51}. Then, extension of this and prior techniques are presented, that allow us to find collisions of reduced versions of SHA-1. We give collisions of variants with up to 40 rounds, and show the complexities of longer variants. These techniques show that collisions up to about 53–58 rounds can still be found faster than by birthday attacks.

1 Introduction

The hash function SHA was designed by the National Security Agency (NSA) and issued by NIST in 1993 as a Federal Information Processing Standard (FIPS-180) [3]. A revised version called SHA-1, which specifies an additional rotate operation to the message expansion, was later issued in 1995 as FIPS-180-1 [4]. The revised version is aimed to be a more secure replacement, that improves the security provided by the hash function. No details of the weaknesses found in SHA-0 were provided. In order to refer more clearly to the first version, we denote it as SHA-0, which is a widely used but non standardized name.

* Part of the results of this paper were given by the first author in an invited talk in SAC 2004, Waterloo, Canada.

** Part of this work was done while visiting École normale supérieure, Paris, France.

*** This work was mostly done while the author was at DCSSI Crypto Lab.

† CNRS UMR-8144.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 36–57, 2005.

SHA-0 and SHA-1 are based on the principles of MD4 [5] and MD5 [6]. They take messages of any length (up to 2^{64} bits) and compute 160-bit hash values.

At CRYPTO'98 Chabaud and Joux [2] proposed a theoretical attack on the full SHA-0 with a complexity of 2^{61}. It is a differential attack that uses a weakness of the expansion algorithm of SHA-0. Their attack is faster than the generic birthday paradox attack and partially explain the withdrawal of SHA-0 by NSA. It is interesting to note that they count the complexity in term of the number of message pairs to be tried and not in term of the number of SHA-0 calls. At first, it may seem to be an artificial way to reduce the claimed complexity by 2. However, due to the use of an early abort strategy in the implementation, the effective complexity in term of SHA-0 calls is roughly 1/4 of the announced value. For the sake of clarity, we continue this tradition and announce all the complexity results by giving the average number of necessary message pairs.

In [1] Biham and Chen discussed near-collisions of SHA-0. By using some of the ideas that originally appeared in [2], they showed that in SHA-0 near-collisions are easier to find than full collisions, and proposed an efficient searching algorithm that eliminates the probabilistic behavior of more than 20 rounds of the algorithm, using the notion of *neutral bits*. When applied to the attack of Chabaud and Joux, this improves the complexity by an approximate factor of 32.

In our current research we improve over the results of [1] in several directions: we first present a tool that uses near-collisions in order to find collisions using a multi-block technique. This tool can be used to attack variants that cannot be attacked by the original technique, as well as to reduce complexities of attacking other variants. With some additional refinements, it also improves the attack on full SHA-0, reducing the complexity down to 2^{51}. Then we present our attacks on reduced-round SHA-1, which can find collisions of up to 53–58 rounds faster than the birthday attack, and show new techniques to attack SHA-1.

In parallel to this paper, Rijmen and Oswald also recently studies reduced versions of SHA-1 [9].

This paper is organized as follows: In Section 2 we describe how near-collisions can be used to find collisions by a *multi-block* technique. In section 3, we show how the multi-block technique can be refined in order to work on the full SHA-0, this leads to a full collision on SHA-0 using messages of four blocks. In Section 4 we describe how the attack on SHA-0 is expanded to attack SHA-1. This section presents various attacks on reduced versions of SHA-1, where each attack emphasizes different aspects and techniques. A 34-round SHA-1 collision that can be found with relatively low complexity is introduced. With this reduced version we show how collisions can be found with messages that have only ASCII letters and even messages with some meaningful words. We continue with a collision of 36-round SHA-1 that uses a message of two blocks, where the first block changes the initial value to a value that is convenient for the attack, and the collision is found in the second block. This attack also shows some differences between the attack of SHA-0 and SHA-1, where the non-linearity of SHA is used in the attack. We then discuss how to bypass the consecutive disturbances problem in the IF rounds. The last attack in this section is a two-block collision of 40-round SHA-

1 that uses the same characteristic in both blocks. All the collisions of reduced SHA-1 that we present were found within a few seconds of computation on a PC. Section 5 analyzes the complexity of attacking various reduced versions of SHA-1 with more rounds, and shows that SHA-1 up to 53–58 rounds can be attacked faster than the birthday attack. The assessments are based on the best characteristics we could find for each reduced version. Section 6 summarizes the paper.

Due to lack of space, we removed the descriptions of SHA-0 [3] and SHA-1 [4], and the description of prior techniques related to this paper, e.g., the original technique for analysis of SHA-0 [2], the improved technique and neutral bits [1]. For descriptions of SHA and these techniques, see the respective references. We also shortened the descriptions of some results and removed some detailed explanations about the attack complexity. The full details will appear in the full version of the paper.

Note. This paper is the result of the merge of two papers by non-intersecting groups of authors. The first group consists of the Technion authors, and the other consists of the DGA and PRISM authors. The multi-block technique as a generic tool including the 50-rounds SHA-0 application and the results on SHA-1 are due to the first group. Motivated by their work, the first author within the second group restarted searching on old, non-working results about iterated collisions in SHA-0. It resulted in an improved multi-block cryptanalysis for full SHA-0, which was then ported and optimized for the supercomputer by the other authors within the group.

2 The Multi-block Tool

SHA uses an iterative process in which each block M_j along with an intermediate value h_{j-1} is subjected to a compression function, whose output is the value of the next intermediate value h_j. Previous works on hash functions, and in particular on SHA, use only one block for the attack. Those attacks start with the initial value h_0 and construct a pair of messages M_1 and M_1^* that output the same h_1 to find a collision, or h_1, h_1^* with a small difference h_1' for near-collisions.

The tool we present in this section uses the iterative process of SHA to find collisions. The idea of this technique is to start with a pair of blocks M_1 and M_1^* that create a near-collision h_1', and continue with a construction of a second block. In the first block we base the message on a characteristic that has a zero input difference h_0', and a non-zero output difference h_1', with some message difference M_1'. In the second block we use a characteristic with a non-zero input difference h_1', and a zero output difference h_2'.

The attack proceeds as follows: Given messages M_1, M_1^* that conform to the first characteristic, we receive the pair of intermediate hash values h_1 and h_1^*. Using these values, we search for a second block M_2, M_2^* whose input values are h_1, h_1^*, and which conforms to the second characteristic. Such a pair will then have $h_2' = 0$, which means a collision after the second block.

As a result we succeed in finding a near-collision in the first block, and then finding a second block, constructed in a similar way, but in which each message

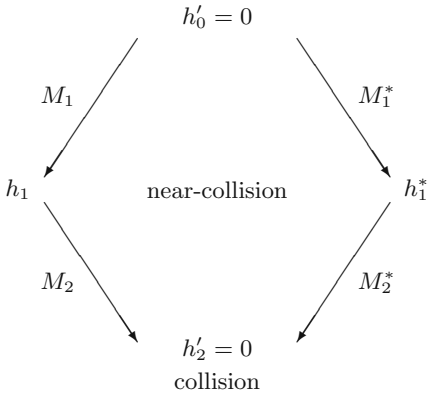

Fig. 1. Using Intermediate Near-Collisions to Find Collisions with Two Blocks

starts with a different input value (rather than same value as is usually done in hash functions) in order to find a collision. An illustration of a two-block attack is given in Figure 1.

The multi-block tool is particularly useful when there is no characteristic that predicts a full collision in one block, and to reduce the complexity of an attack when a single-block collision is more complex.

It should be noted that Wang [7, 8] independently used two message blocks to find the collision of MD5, using a first block that creates a near-collision, and a second block that restarts from this near-collision and ends with a collision.

Applications. In order to illustrate the multi-block technique, we can apply to SHA-0 reduced to 50-rounds. This example is interesting, since this reduced version does not have any characteristic (i.e., any disturbance vector) that predicts a collision with a single block. However, it is very easy to find near-collisions with complexity of about 2^{17}. Using the multi-block technique, we can restart from this near-collision in order to find a longer message pair that collides after the second block. The total complexity remains about 2^{17}.

Collisions with More than Two Blocks. This technique can be generalized to several blocks. In the case of two blocks the first block of the messages M, M^* is constructed by using a characteristic that has a zero input difference h'_0 and, a non-zero output difference h'_1. In the second block we use a characteristic whose input difference is h'_1, and which has a non-zero output difference h'_2. In the case of two blocks $h'_2 = 0$, which means a collision. However, in case $h'_2 \neq 0$, it is possible to use h'_2 as the input difference of a third block which leads to a collision (see Figure 2). Alternatively the third block can lead to another near-collision that may later be converted to a collision of the fourth block. In general the technique can find k-block collisions, where the first block starts with $h'_0 = 0$, with $k - 1$ intermediate near-collisions $h'_i \neq 0$ $(i = 1, \ldots, k - 1)$, which lead

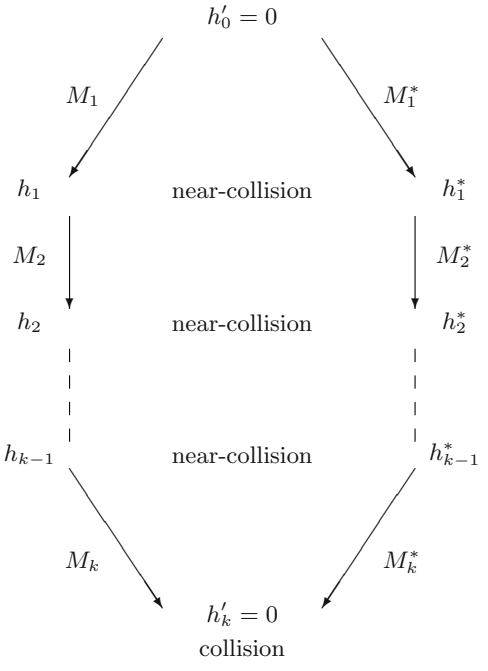

Fig. 2. The Multi-Block Technique—Using Intermediate Near-Collisions to Find Collisions

to a collision with $h'_k = 0$ after k blocks. The complexity of finding the k-block collision is the sum of the complexities of finding the $k-1$ near-collisions and the final collision. More information on usage of multi-block collisions will appear in the full paper.

3 A Multi-block Collision of SHA-0

Since the multi-block technique described above is very promising, it is extremely tempting to apply it to the full 80 rounds SHA-0. Unfortunately, contrarily to what happens with the 50-rounds version, there is no attack of this type which behaves better than the single block attack proposed by Chabaud and Joux. All the other paths that use near collisions happen to be dead-ends.

In order to remove this obstruction, another key idea is necessary. We should note that in the early rounds of SHA-0, an IF function is used. This means, that during the early rounds, SHA-0 may in some case behave differently than the linearized model of [2]. This misbehavior might allow us to connect differentials which do not belong together in the linearized model of SHA-0. In order to make this idea precise, we first introduce some notations to describe the differences before and after each block. First, remark that in each register A to E, after

a successful application of a one block differential, a difference may occur at a single, fixed, position. In A and B a difference may occur at bit 1, in C, D and E at bit 31. As a consequence, to describe an initial or final difference, a 5-bit number suffices. We assign the high order bit to A and the low order bit to E. Thus, a state with a single difference D will be referred to as state 2. The second step is to compare the expected behavior of a reference state in the linearized model with the possible behaviors of a given state when the IF function is used, i.e., in real-life SHA-0. This is done by examining how the initial difference propagates in the five first rounds.[1] To start with a simple example, assume that reference state 2 is considered in the linearized model. In that case, we have a single initial difference on bit 31 of D. Due to the XOR function, this difference propagates in the update formula for the next value of A. Thanks to the disturbance vector, it will be adequately corrected, however, this is not relevant for this part of the discussion, we just need to know that it propagates in the formula. Then, the registers are shifted and the initial difference moves to E. In the next update formula, it will also propagate, again on bit 31. After that round, the initial difference has vanished and no longer propagates. Now, consider that state 3 enters the real SHA-0. Then, in the first formula, both D and E have a difference on bit 31, however, depending on the result of the IF function the difference on D may either propagate or not. More precisely, if bit 31 of register B (which is the same in both messages) is a 1, the difference on D does not propagate. On the other hand, the difference on E always propagates. The gross result is that a single difference propagates on bit 31, thus at this point state 3 behaves as reference state 2 in the linearized model. After the registers shift, a difference remains on E and it propagates in the second update formula. As a consequence, we see that real state 3 may behave as reference state 2. Thus, we may start a differential attack from state 3 by using a disturbance vector that "expects" state 2. Moreover, state 3 may also behave like reference state 3. This implies, that it is possible to connect together much more differentials than initially expected. Thus, the graph of possible paths is considerably richer than first predicted and we expect to find a better attack.

With this translation table in mind, we now try to assemble several differentials with different disturbance vectors into a global attack. For any disturbance vector, we add five extra bits, the "negative" bits which indicate the starting reference state. Similarly, the value of the last five bits indicate the expected state after the block cipher part of the compression function. To incorporate such a disturbance vector into the global attack, we proceed as follows: Assume that the current state is a and that we are given a disturbance vector $a' \rightarrow b$, i.e., a disturbance that goes from reference state a' to expected state b, then if a

[1] We further remark that this representation can be extended to a general kind of characteristics describing the evolution of differences in registers $A, \ldots, E$, and in the expanded message, in a similar way to the characteristics used in related-key differential cryptanalysis. In such a case, the intermediate differences can be very different than predicted by the model of [2], while still leading to collisions.

is compatible with a', we have a differential that goes from state a to next state $a \oplus b$ after the final addition. Thus, we can build a transition graph, where each possible state is a node, and each differential, with good enough probability, is an edge. In this graph, we now search for a path from state 0 to itself, with low expected complexity. The best path we could find has length 4, it starts from state 0, goes to 3, 25, 8 and finally comes back to zero. It is build on the following disturbance vectors:

Ref (DV) States	DV		Actual States	Compatible With
$0 \rightarrow 3$	00000 0001000010100100011	10010110000011100000	$0 \rightarrow 3$	**2** 3
	00000011000000110110	00000110001011011000		
$2 \rightarrow 26$	01000 1000000010000101001	00011110010110000011	$3 \rightarrow 25$	**17** 18 28 31
	10000000000011000000	11011000000110001011		
$17 \rightarrow 17$	10001 0010010100010010111	11000010000100001100	$25 \rightarrow 8$	8 **11** 13 14
	00101100100000000001	11010011101000010001		
$11 \rightarrow 8$	11010 0010000000100001010	01000111100101100000	$8 \rightarrow 0$	collision
	11100000000000110000	00110110000001100010		

One can easily check that this sequence of block differences can possibly lead to a full collision. Initially, the difference between the two messages of a pair corresponds to state 0. After the first block, we intend to reach state 3. Of course, for this block the final additions add equal values on each branch, thus the difference is expected to remain at state 3. Since state 3 is compatible with reference state 2, we restart from there and go to state 26. For this second block, the additions change state 26 into state 25. Again, the compatibility of 25 with reference state 17 allows us to restart with the third block difference. The expected state is 17 before the final additions and 8 after them. Thanks to the compatibility of 8 with 11, we use the fourth difference and expect a state of 8 before the additions. Since the two states 8 correspond to differences on the same bits, we expect that they cancel each other. Thus, we finally reach a full collision. Evaluating the exact complexity of this attack requires a detailed analysis that, for lack of space, is not given here. The total cost is 2^{51} message pairs, as confirmed by our implementation.

3.1 Implementation and Optimization

The theoretical complexity of our collision search algorithm is 2^{51}. This complexity is expressed in term of the number of pairs of messages to test. As is the case with the original attack of Chabaud and Joux, this roughly corresponds to the cost of 2^{49} evaluations of the SHA-0 compression function.

In order to demonstrate feasibility of this collision search, we implemented this algorithm on an Intel Itanium 2 processor. This processor allows a wide degree of instruction level parallelism (ILP). More precisely, it is able to execute up to six instructions per cycle, and a wide variety of combinations is possible (e.g., 6 arithmetical operations, or 4 memory operations and 2 floating point multi-

ply add, or 3 logical operations and 3 branches, etc.). Furthermore, this wide ILP capability is enhanced by a large register file and many duplicated functional units. The processor also offers several mechanisms to implement control/branch structures with speculative execution, predication, and multi-way branches (up to three branches per cycle). Due to the complex nature of the processor, the performance of programs running on it heavily relies on the capability of the compiler to produce efficient code. Our algorithm was compiled by the Intel compiler (ICC) whose performance in this respect is usually above average.

To optimize our code, a *profiling* step was performed to detect the most time-consuming code sections. This study revealed that the main function, which enumerates pairs of messages derived from a reference pair and its' neutral set represented a large majority of the execution time. Focusing on this part, we checked the behavior of the code at the hardware level during execution through the use of *hardware counters*.

We, thus, determined that the main performance limiting factors were:

- *Limited amount of parallelism*: All rounds of SHA-0 contain chains of bitwise operations ($+$, ROL_x, ...) depending on each other, which limited the effect of the internal parallelism.
- *Complex control flow*: Due to the probabilistic nature of the collision search, the control flow is quite complex and statistically (at compile time) unpredictable.
- *Cached memory access*: Despite being in a very favorable case where all data fits in the first level data cache (L1D) of the Itanium 2 (16KB), the number of accesses to the cached memory is very high, when arrays are used to represent the intermediate values during the computation. As a consequence, memory access in L1D was a bottleneck in our basic implementation.

Due to the complex control flow, the Intel compiler could not determine a good way to execute branches. Even the use of advanced optimization tools such as *profile guided optimization*, did not help much. The compiler still used speculative execution, which led to bad performance. A first step in our tuning process was to make the compiler avoid speculation, by writing each round differently, depending on the probability of success at this point.

Since the number of L1D memory accesses was critical, the second step consisted in reducing them. This was done by replacing all arrays by registers thus avoiding many memory stores and loads. This optimization makes good use of the large number of registers of the Itanium 2. Such a technique is called *register promotion* and is usually performed by the compilers. However, in this example, this had to be done on a large number of source lines and the compiler was unable to deal with this. Moreover, we had to extend the technique to deal with the complex control structure.

All the fine tuning techniques allowed to gain an additional 20% of performance compared to the best compiler options (which are not the standard `O3` options and had to be determined through exhaustive search). On average, 4 instructions per cycle were effectively executed, out of a maximum 6.

3.2 A Full Collision of SHA-0

Once the program was ported to the supercomputer, it processed a large number of messages pairs for each block. Very precisely, the total number of trial pairs was:

$$\begin{array}{ll} \text{First block} & 796\,682\,307\,091\,035 \approx 2^{49.5} \\ \text{Second block} & 1\,572\,177\,940\,314\,628 \approx 2^{50.5} \\ \text{Third block} & 1\,712\,558\,626\,669\,268 \approx 2^{50.6} \\ \text{Fourth block} & 17\,049\,400\,703\,749 \approx 2^{44} \end{array}$$

We can remark that the number of computations is higher than expected for the first two blocks. At first, we simply assumed that we had been unlucky, however, a deeper investigation revealed a subtle bug in the neutral bits identification code. Due to this bug, some messages pairs were processed more than once, and up to four times, by the program. These useless computations explain the mismatch between the predicted complexity of the first two blocks and the effective numbers of messages pairs processed. Luckily, the bug did not affect the computation of block 3, thus the total slowdown was limited. Finally, we reached the following messages (written in hexadecimal):

```
a766a602 b65cffe7 73bcf258 26b322b3 d01b1a97 2684ef53 3e3b4b7f 53fe3762
24c08e47 e959b2bc 3b519880 b9286568 247d110f 70f5c5e2 b4590ca3 f55f52fe
effd4c8f e68de835 329e603c c51e7f02 545410d1 671d108d f5a4000d cf20a439
4949d72c d14fbb03 45cf3a29 5dcda89f 998f8755 2c9a58b1 bdc38483 5e477185
f96e68be bb0025d2 d2b69edf 21724198 f688b41d eb9b4913 fbe696b5 457ab399
21e1d759 1f89de84 57e8613c 6c9e3b24 2879d4d8 783b2d9c a9935ea5 26a729c0
6edfc501 37e69330 be976012 cc5dfe1c 14c4c68b d1db3ecb 24438a59 a09b5db4
35563e0d 8bdf572f 77b53065 cef31f32 dc9dbaa0 4146261e 9994bd5c d0758e3d
```

and

```
a766a602 b65cffe7 73bcf258 26b322b1 d01b1ad7 2684ef51 be3b4b7f d3fe3762
a4c08e45 e959b2fc 3b519880 39286528 a47d110d 70f5c5e0 34590ce3 755f52fc
6ffd4c8d 668de875 329e603e 451e7f02 d45410d1 e71d108d f5a4000d cf20a439
4949d72c d14fbb01 45cf3a69 5dcda89d 198f8755 ac9a58b1 3dc38481 5e4771c5
796e68fe bb0025d0 52b69edd a17241d8 7688b41f 6b9b4911 7be696f5 c57ab399
a1e1d719 9f89de86 57e8613c ec9e3b26 a879d498 783b2d9e 29935ea7 a6a72980
6edfc503 37e69330 3e976010 4c5dfe5c 14c4c689 51db3ecb a4438a59 209b5db4
35563e0d 8bdf572f 77b53065 cef31f30 dc9dbae0 4146261c 1994bd5c 50758e3d
```

which have the same hash values. More precisely, the intermediate hashes for both messages are compatible with the predictions of our differential attack and their precise values are:

IV	67452301	EFCDAB89	98BADCFE	10325476	C3D2E1F0
Block 1	83C1CE2D	C5BF5480	C2AF2358	104B337B	9E78A1E7
Block 2	27AE025A	9D36F7B6	29FA88E7	87B70063	984119F3
Block 3	4DD120B4	D6EC801F	468628A7	0CC26464	371F36B2
Block 4	81FB4643	08FDF1F4	A3C4F3A3	6188FED3	FD2378E6
Padding	C9F16077	7D4086FE	8095FBA5	8B7E20C2	28A4006B
IV	67452301	EFCDAB89	98BADCFE	10325476	C3D2E1F0
Block 1	83C1CE2D	C5BF5480	C2AF2358	904B337B	1E78A1E7
Block 2	27AE0258	9D36F7B4	29FA88E7	87B70063	184119F3
Block 3	4DD120B4	D6EC801D	468628A7	0CC26464	371F36B2
Block 4	81FB4643	08FDF1F4	A3C4F3A3	6188FED3	FD2378E6
Padding	C9F16077	7D4086FE	8095FBA5	8B7E20C2	28A4006B

In this table, the underlined values highlight the difference between the two hash processes. These differences are as predicted by our differential attack. After the fourth blocks, the two messages collide. Of course, since the two messages have the same length, the padding blocks are identical. Thus, the final values inherit from the fourth block collision.

4 SHA-1 Results

Our attack on SHA-1 extends the techniques of [1] designed for SHA-0. The only difference between SHA-1 and SHA-0 is an additional rotation operation in the expansion process. Due to this rotation SHA-1 mixes the bits in the expanded message in a more efficient way than SHA-0 does, thus making the attack much less efficient against SHA-1 (as was already noted in [1]). In this section we observe that with some modifications, the attack can be applied to reduced versions of SHA-1. In the next subsections we present collisions of 34–40 rounds SHA-1 that we found using this application.

4.1 A Collision of 34-Round SHA-1

The attacks of SHA-0 use only bit 1 as the location of disturbances. This bit is selected to eliminate the probabilistic behavior of the carry when corrections are applied to bit 31, thus increasing the total probability of the characteristic. Since the expansion process in SHA-0 does not mix bits in different locations in the 32-bit word, all the disturbances in the expanded message are in bit 1, but this is not the case in SHA-1. Therefore, other bits can be used as disturbances. With this change in the selection, a disturbance vector in SHA-1 is not boolean, in which each entry tells whether there is a disturbance in bit 1, but instead a 32-bit word that represents all the disturbances in a round. Following this change, the corrections associated with a disturbance vector are derivated slightly differently than in [1] (i.e., corrections are applied to each disturbance relative to its location).

We observe that for 34-round reduced SHA-1 (unlike longer versions) there is a disturbance vector with a very low Hamming weight, which is given in Table 1. In this table D.Vec column shows the expected values of A'_{i+1} for $i = 0, \ldots, 33$,

Table 1. The Disturbance Vector Used for 34-round SHA-1 (in 32-bit hex words)

Rnd	D.Vec	D&C	Rnd	D.Vec	D&C	Rnd	D.Vec	D&C
−5	00000000		8	00000000	80000003	21	00000000	00000040
−4	00000000		9	00000002	40000002	22	00000002	00000000
−3	00000000		10	00000000	C0000040	23	00000000	80000040
−2	00000000		11	00000000	C0000002	24	00000000	80000002
−1	00000000		12	00000000	80000000	25	00000000	00000000
0	00000002	00000002	13	00000000	80000000	26	00000000	80000000
1	00000000	00000040	14	00000002	80000002	**27**	**00000000**	80000000
2	00000002	00000000	15	00000000	00000040	**28**	**00000000**	00000000
3	00000000	80000040	16	00000000	00000002	**29**	**00000000**	00000000
4	00000002	80000000	17	00000000	80000000	**30**	**00000000**	00000000
5	00000000	00000040	18	00000000	80000000	**31**	**00000000**	00000000
6	00000003	80000001	**19**	**00000000**	80000000	**32**	**00000000**	00000000
7	00000000	00000060	20	00000002	00000002	**33**	**00000000**	00000000

Table 2. Collision of SHA-1 Reduced to 34 Round (in 32-bit hex words)

Message 1:
`F1641C2B 242BFDB5 EAE01E30 F4BBBA6F 18D45E8E DE68AEBA 74EC8CF9 FC204957`
`45AAA8BF 1CD3AE7D D845C2F3 AC737464 F25BEBBB BE5FFF1D 2ADB2818 0B1D13FB`
Message 2:
`F1641C29 242BFDF5 EAE01E30 74BBBA2F 98D45E8E DE68AEFA F4EC8CF8 FC204937`
`C5AAA8BC 5CD3AE7F 1845C2B3 6C737466 725BEBBB 3E5FFF1D AADB281A 0B1D13BB`

which we denote by δ_{i+1} (note that the indices of δ here are 1, ..., 34, rather than 0, ..., 33, as δ_i is the difference at the input of round i). The D&C column shows the message difference (which is underlined and shown in rounds $0, \ldots, 15$), and the differences of the expanded messages in rounds $16, \ldots, 33$. $\delta_{-4}, \ldots, \delta_0$ are also shown in this table (in rounds $-5, \ldots, -1$), and their values are related to the initial value differences. Intermediate rounds after which collisions are predicted are marked in boldface. This disturbance vector is unique in that almost all the disturbances are in one location (bit 1), and in all of the 34 rounds there is only a single disturbance in a different location, which is bit 0 in round 6. This disturbance succeeds to cancel the avalanche effect that is expected in SHA-1 due to the additional rotate operation, and that does exist in other disturbance vectors.

By using the neutral bits method of [1], the complexity of an attack can be estimated based on the number of disturbances after round 20. Thus, using this disturbance vector, that has only two disturbances after round 20, we easily found millions of collisions, one of which is given in Table 2.

Since the complexity of finding collisions in this 34-round attack is so low, we were able to generate collisions with additional constraints, which caused some increase in the complexity. This way we found collisions whose all bytes

Table 3. Two Messages in ASCII Letters that Collide Under 34-Round SHA-1

Message 1:
IkGDqVMwISGGcBMpNHMYavPTsmUlykPTzokJOkwnrSgJSfDmlpeqsmDzWbAjmNxP
Message 2:
IkgDqRMwISGGcFEpNHEYarPTsmMlymPTzoSJOkSnrWkJSfhmlpmqsmLzWbijmJxP

Table 4. Two Examples of Partially Meaningful Messages that Collide Under 34-Round SHA-1

Message 1:
I Am OilMANgujnPay916472136314$USAkNOWwTkjepMFXGlmfHNGcpodElGfvL
Message 2:
I am KilMANgunfPay11607213.312$USASNOWSTknipMFtGlmnHNGkpodmlGbvL
Message 1:
OhG, not this mess,age notThat onenot U, oh noHRtBMTkKllLlIluvpB
Message 2:
Ohg, jot this$eess$aga notLhar oneVot q, kd nodRtBETkKdlLlalurpB

are formed of ASCII letters. The disturbance vector of Table 2 does not allow that, as some bytes of the message differ in the most significant bits. However, by rotating the locations of the disturbances (by the same number of bits in all the rounds) we can move the differences to lower bits, while increasing the complexity of the attack by a small factor. A colliding pair of messages consisting entirely of letters in ASCII is given in Table 3.

With some additional creativity, and some additional increase in the complexity, it was also possible to force some of the bytes into partial English text. Table 4 lists two examples. The first example is an attempt to force the two colliding messages to contain meaningful text. However, there are still many constraints on the possible text, thus it can be seen that some letters are capitalized, while other are not, that some spaces appear between words, while they do not appear between other words, and that some random letters must be allowed in some locations in order to allow more text afterwards. The second example in Table 4 is an attempt to further improve the text of one message in the expense of the text of the other message.

4.2 A Collision of 36-Round SHA-1

In this section we present a collision of 36-round reduced SHA-1 along with several techniques that were used to find it.

In our attack on 36-round SHA-1 we use the best characteristic that predicts a collision after one block. We show this disturbance vector in Table 5.

It should be noted that this characteristic cannot be used with the standard initial value of SHA-1, i.e., with

$$h_0 = (67452301_x,\ EFCDAB89_x,\ 98BADCFE_x,\ 10325476_x,\ C3D2E1F0_x),$$

Table 5. The Disturbance Vector Used for 36-Round SHA-1 (in 32-bit hex words)

Rnd	D.Vec	D&C	Rnd	D.Vec	D&C	Rnd	D.Vec	D&C
−5	00000000		9	00000002	00000008	23	00000000	80000050
−4	00000000		10	00000002	00000042	24	80000003	80000001
−3	00000000		11	80000000	50000042	25	00000000	A0000070
−2	00000000		12	00000002	10000010	26	00000000	20000003
−1	00000000		13	00000000	90000040	27	00000002	40000002
0	80000000	80000000	14	00000002	20000000	28	00000000	E0000040
1	00000000	00000010	15	00000000	20000040	29	00000000	E0000002
2	00000001	80000001	16	80000003	20000001	30	00000002	80000002
3	00000000	20000020	17	00000000	00000070	31	00000000	80000040
4	00000003	20000002	18	00000000	00000003	32	00000000	80000002
5	00000002	60000062	19	00000002	60000002	33	00000000	80000000
6	00000001	40000042	20	00000000	E0000040	34	00000000	80000000
7	00000002	80000020	21	00000000	E0000002	35	**00000000**	80000000
8	40000000	00000041	22	80000002	00000002			

Table 6. The Second Block of the Collision of 36-Round SHA-1 (in 32-bit hex words)

Common block 1: sixteen 00000000 words
Message 1, block 2:
9F29DE8D BBD58270 1F11EB22 A6637C3E 7E6FB0C0 63E9BF5E C4FF7010 073174B3
3133689A 579A753E 2D17124D 7D37E853 5B5BBB01 F0371FBB 025A725C 8FB9FE33
Message 2, block 2:
1F29DE8D BBD58260 9F11EB23 86637C1E 5E6FB0C2 03E9BF3C 84FF7052 87317493
313368DB 579A7536 2D17120F 2D37E811 4B5BBB11 60371FFB 225A725C AFB9FE73

due to the observation that in round 2 there is a difference in the most significant bit of register B ($B' = 80000000_x$), but both most significant bits of C and D are zero (where $C = 67452301_x \lll 30$ and $D = EFCDAB89_x \lll 30$). Thus, considering the differences of the messages ($W_2' = 80000001_x$) in that bit, the new content of A must have a difference in this bit, in contrary to the prediction of the disturbance vector.

In order to be able to use the disturbance vector of Table 5, the initial value is replaced by another value by adding an additional first block, which in this case is the whole zero block ($M_1 = M_1^* = 0$). The resultant intermediate hash value is

$$h_1 = (37970DFF_x,\ 5E912289_x,\ C78B3705_x,\ 923B82E9_x,\ CC36E948_x).$$

With this intermediate value h_1, we can now proceed to the next block with the disturbance vector of Table 5. The second block of the collision of the 36-round SHA-1 is presented in Table 6.

Table 7. Comparison of δ_i and A'_i in the 36-Round Collision (in 32-bit hex words)

Round	$D\&C$	δ_{i+1}	A'_{i+1}	B'_{i+1}	C'_{i+1}	D'_{i+1}
0	80000000	80000000	80000000	00000000	00000000	00000000
1	00000010	00000000	00000000	80000000	00000000	00000000
2	80000001	00000001	00000001	00000000	20000000	00000000
3	20000020	00000000	00000000	00000001	00000000	20000000
4	20000002	**00000003**	**00000001**	00000000	40000000	00000000
5	60000062	00000002	00000002	00000001	00000000	40000000
6	40000042	00000001	00000001	00000002	40000000	00000000
7	80000020	00000002	00000002	00000001	80000000	40000000
8	00000041	40000000	40000000	00000002	40000000	80000000
9	00000008	00000002	00000002	40000000	80000000	40000000
10	00000042	00000002	00000002	00000002	10000000	80000000
11	50000042	80000000	80000000	00000002	80000000	10000000
12	10000010	00000002	00000002	80000000	80000000	80000000
13	90000040	00000000	00000000	00000002	20000000	80000000
14	20000000	00000002	00000002	00000000	80000000	20000000
15	20000040	00000000	00000000	00000002	00000000	80000000

A Generalized Test for Conformance. The 36-round collision of Table 6 presents an additional change in respect to the attack of SHA-0. In the attack on SHA-0, the intermediate differences A'_i are necessarily equal to δ_i for $i = 1, \ldots, r$, where r is the number of rounds of the analyzed compression function. In SHA-1 this is not the case, since more than a single location of a bit are selected for the disturbances. In cases where there are two or more disturbances or corrections in adjacent bits, it may happen that the more significant bit is not correctly approximated, e.g., the IF function does not output the XOR of its inputs for the particular values of the registers. However, it may happen that the carry of the less significant bit cancels this wrong approximation, resulting with the expected difference $A'_i = \delta_i$. In other cases, a wrong approximation of the less significant bit cancels the correct approximation of the more significant bit, e.g., the addition modulo 2^{32} of the less significant bit changes the carry. In these cases $A'_i \neq \delta_i$, and the difference is in this more significant bit. The difference that the more significant bit expects to create in A'_i is now canceled, but the corrections for this expected difference still exist in the following five rounds. These corrections are now used to correct wrong approximations of the less significant bit which change the carries in the next five rounds. If we are lucky, the less significant bit creates additional differences in the carry, thus corrects the differences in A'_i in the next rounds.

Table 7 shows the differences of first 16 rounds of the compression function in the second block of the 36-round collision (shown in Table 6). In this table the D&C column shows the message difference M', δ_{i+1} shows the expected difference in A'_{i+1}, and the other four columns show the actual difference A'_{i+1}, B'_{i+1}, C'_{i+1}, and D'_{i+1}. The table shows a situation where two disturbances are applied to bit 0 and 1, and the carry change of bit 0 cancel the disturbance of

bit 1. The entry of round 4 in the table shows (in boldface) that the expected difference δ_5 is different from the actual value of A'_5. This difference between the expected and actual values is due to a carry change of the disturbance of bit 0 that cancels the difference in bit 1. The five corrections in the next five rounds do not have a disturbance in registers A, B, C, D, nor E, but other properties of the IF and carry overcome the missing difference and ensure correct differences in the following rounds.

We call bits whose difference may differ from the expected value of the characteristic, but whose effect can be canceled immediately afterwards, by the name *T bits*. In some cases a simultaneous modification of a few bits makes a similar effect. We can view T bits as extending the notion of characteristics into differentials in which most information on the intermediate differences is fixed, but a few can have any value, describing several different paths leading to the same differential. There are several T bits in the intermediate differences characteristic of 36-round SHA-1, and also in other characteristics used in this paper.

Due to such cases we extended our program to check for conformance by testing for a generalized kind of differences instead of testing exactly whether $A'_i = \delta_i$.

Consecutive Disturbances in the IF Rounds. In the attack on SHA-0 two consecutive disturbances in the first 17 rounds (i.e., rounds $0, \ldots, 16$) have a probability zero to be corrected (see [2]). This limitation forces a higher Hamming weight to occur in the expanded disturbance vector, but an attack is still feasible (i.e., there are still few disturbance vectors that predict collisions, and do not have two consecutive disturbances in the first 17 rounds). We observed that all the disturbance vectors that we could find that predict one-block collisions of SHA-1 reduced to 35 or more rounds have consecutive disturbances, i.e., two disturbances at the same bit locations in two consecutive rounds. Thus, this limitation seems to be much more restrictive in SHA-1. However, this stronger limitation comes with the ability to bypass it by various techniques in some fraction of the cases. The characteristic we use for the collision of 36-round SHA-1 is an example for such a case.

In the following discussion, we first explain the limitation of the two consecutive disturbances in SHA-0, and then we show how they behave in SHA-1. In SHA-0, two consecutive disturbances in rounds i and $i+1$ (in bit 1) create differences in $D^{31'}_{i+4}$ and $C^{31'}_{i+4}$, respectively. The two corrections to these differences are applied to the same bit, thus cancel each other in the approximation leading to no difference in δ_{i+4}. On the other hand, the IF function applied on these two differences, where the difference of $B^{31'}_{i+4}$ is zero, causes the result to be complemented always. Thus, in A'_{i+4} we have a difference with no corrections. With SHA-1 the same arguments apply, but we allow disturbances at any bit location. Thus, we can use the carry bit from another disturbance (or correction) as an additional source of corrections.

The following two examples, which are taken from our 36-round attack, should clarify the above: In the first example we show how a carry can be used as follows: At rounds 4 and 5 there are disturbances in bit 1, from which we

expect to get A'_5 and A'_6 equal to $\delta_5 = 00000003_x$ and $\delta_6 = 00000002_x$ respectively, which lead after three rounds to the differences $D'_8 = C0000000_x$ and $C'_8 = 80000000_x$. With these differences the IF function applied on $D_8^{31'}$ and $C_8^{31'}$ always complement the output, but it is never complemented in the approximation. Thus, we have a difference that cannot be corrected. However, in the messages we use the carry from bit 0 at round 4 cancels the disturbance at bit 1 of this round, and therefore the created differences are $A'_5 = 00000001_x$ and $A'_6 = 00000002_x$ (see Table 7). Thus, in round 8 the differences are $C'_8 = 80000000_x$ and $D'_8 = 40000000_x$, which can be corrected by the non-linear behavior of the IF function to fit the approximation.

In the second example we show how the problem of two consecutive disturbances can be bypassed when there is another disturbance in one of a few different locations. In rounds 9 and 10 (see Table 7) we have two consecutive disturbances in bit 1 ($\delta_{10} = 00000002_x$ and $\delta_{11} = 00000002_x$), but in this case there is also a disturbance in round 11 in bit 31 ($\delta_{12} = 80000000_x$). Thus, in round 13 we have $B'_{13} = C'_{13} = D'_{13} = 80000000_x$, which fit the approximation with probability 1/2.

In general, consecutive disturbances in bit j of rounds i and $i+1$ can be corrected, if there is a correction or disturbance in a less significant bit that may change the carry to bit $j-2$ in round $i+4$ (i.e., in bit $j-8$ of δ_{i+3}, bit $j-1$ of δ_{i+2}, or bit $j-1$ of δ_{i+1}, δ_i or of δ_{i-1} where the bit numbers are mod 32), leaving the rest of the differences behave as expected.

4.3 A Two-Block Collision of 40-Round SHA-1

In this section we present a collision of 40-round reduced SHA-1. The best (one-block) characteristic that we could find has 19 disturbances from round 20 to round 39, so the complexity of the attack is expected to be around 2^{57}. However, it is easy to find near-collisions of 40 rounds with only five disturbances from round 20 to 39. Thus, we construct a two-block attack where the first block generate such a near-collision, and the second block uses the difference of the initial value that are created by the first block and generate a collision.

We observe that the hash values of multi-block messages are computed as the sum of the initial value and the states g_i of the compression function before the final addition operations, i.e.,

$$h_n = h_0 + \sum_{i=1}^{n} g_i.$$

Therefore, for colliding pairs of messages the following equation holds

$$\sum_{i=1}^{n} (g_i - g_i^*) = 0,$$

which when the addition is approximated by XOR becomes

$$\sum_{i=1}^{n} g'_i = 0.$$

Table 8. The Disturbance Vector Used for the Two-Blocks Collision of 40-round SHA-1 (in 32-bit hex words)

Rnd	D.Vec	D&C	Rnd	D.Vec	D&C	Rnd	D.Vec	D&C
−5	00000000		10	00000000	0C000004	25	00000000	00000000
−4	00000000		11	00000000	2C000000	26	00000000	08000000
−3	00000000		12	00000000	08000000	27	00000000	08000000
−2	00000000		13	00000000	08000000	28	00000000	00000000
−1	00000000		14	20000000	28000000	29	00000000	00000000
0	20000000	20000000	15	00000000	00000004	30	00000000	00000000
1	00000000	00000004	16	00000000	20000000	31	00000000	00000000
2	20000000	00000000	17	00000000	08000000	32	00000000	00000000
3	00000000	08000004	18	00000000	08000000	33	00000000	00000000
4	20000000	08000000	19	00000000	08000000	34	40000000	40000000
5	00000000	00000004	20	20000000	20000000	35	(00000000)	00000008
6	30000000	18000000	21	00000000	00000004	36	(00000000)	40000000
7	00000000	00000006	22	20000000	00000000	37	(80000000)	90000000
8	00000000	38000000	23	00000000	08000004	38	(40000000)	50000010
9	20000000	24000000	24	00000000	28000000	39	(00000000)	90000008

Therefore, when searching for multi-block collisions it may be best to find characteristics for which this sum is zero, and verify that all the other requirements are satisfied, rather than vice versa.

In the particular case of a two-block collision this equation means that $g'_1 = g'_2$, i.e., the two disturbance vectors should have same differences in the last five rounds. This leads to the question why should we use different disturbance vectors for both blocks. The answer would be that the initial value difference of the second block is necessarily different than of the first block (as $h'_0 = 0$ and $h'_1 \neq 0$), where the initial value is related to the difference of the first five rounds of the disturbance vector (rounds $-5, \ldots, -1$). But this is only a partial answer, as we can extend the technique (using for example T bits, with similarities to the extension of Section 3 in the case of SHA-0, but with much more flexibility), and use a disturbance vector whose first five rounds are different than the initial value difference (in the second block). Once we say that, we observe that in the case of the disturbance vector that we use for the first round, the intermediate value h'_1 fits as a replacement initial difference for the same disturbance vector, i.e., if we replace rounds $-5, \ldots, -1$ of the disturbance vector by the last five rounds from the first block, we still get differences that can be corrected later by the disturbance vector. In terms of characteristics, this means that we have two characteristics with different input differences, but same message differences and output differences (and that in most of the rounds they have the same intermediate differences).

Table 8 describes the disturbance vector we use for this attack. This disturbance vector is the same vector used in our 34-round collision (Table 1) rotated by 28 bits to the left and expanded to 40 rounds. In the first five rounds ($-5, \ldots,$

Table 9. The Beginning of Both Blocks of the Disturbance Vector Used for 40-round SHA-1 (in 32-bit hex words)

Round	First Block D.Vec	Second Block D.Vec	Common D&C
−5	00000000	(00000000)	
−4	00000000	(00000000)	
−3	00000000	(80000000)	
−2	00000000	(40000000)	
−1	00000000	(00000000)	
0	20000000	20000000	20000000
1	00000000	00000000	00000004
2	20000000	20000000	00000000
3	00000000	00000000	08000004
4	20000000	20000000	08000000
5	00000000	00000000	00000004
6	30000000	30000000	18000000
7	00000000	00000000	00000006
8	00000000	00000000	38000000
9	20000000	20000000	24000000
⋮	⋮	⋮	⋮

−1) of the disturbance vector the differences are zero, and in the last five rounds they have two active bits (these rounds are marked in parentheses). Therefore, we expect that h'_1 will have two active bits in these locations (up to the rotation by 30 bits), so the disturbance vector for the next block should have the first five rounds with the same differences as given in parentheses in the table. Now, we observe that when we replace the first five rounds of the same disturbance vector with the values in parentheses (see Table 9) we still receive a correctable result. The disturbance vector itself, from round 0 to round 39 is unchanged, thus the modified five rounds do not fit to the expansion function of SHA-1, but as these difference come from the initial value, they are not calculated anyway by this expansion. These values should only ensure that the probability of the rounds in which they participate (as A, B, C, D, or E) is greater than zero, and this is the case with these replaced differences.

We would also wish to add that the change of the initial rounds of the disturbance vector can be even extended to a few additional rounds, as long as the message differences remain unchanged, i.e., it would be possible to expect for different values in round 0 (or even 1) of the disturbance vector when changing the initial five rounds, but without changing the message differences. Also, it is possible to make replacements in the last few rounds. This phenomena is similar to the usual technique of differential cryptanalysis, where iterative characteristics are used with modified first and last rounds, allowing even larger probabilities than in the full iterative case.

The messages of the 40-round collision are presented in Table 10. The output difference h'_1 of the compression function of the first block becomes the input

Table 10. The Two-Block Collision of 40-Round SHA-1 (in 32-bit hex words)

Message 1, block 1:
404B674C B70CB385 D2DDAC0D 3A0E9BD3 CA7F1780 7FEFDA17 05E43AF2 444344C2
641A2CB6 86C2CFE6 EBCDEF67 6577E095 1A9CAD10 CFE48484 78639157 B13B759A
Message 2, block 1:
604B674C B70CB381 D2DDAC0D 320E9BD7 C27F1780 7FEFDA13 1DE43AF2 444344C4
5C1A2CB6 A2C2CFE6 E7CDEF63 4977E095 129CAD10 C7E48484 50639157 B13B759E
Message 1, block 2:
E63C47F7 0AB5F259 47DE1E6B 09E06877 6229CC42 604CF1AB 9B14B8F3 7261186C
1A5370F9 822E13EB FB7157EF 6B0919C5 1F3D744B FA4DE198 FBB10C06 FDA3C3E9
Message 2, block 2:
C63C47F7 0AB5F25D 47DE1E6B 01E06873 6A29CC42 604CF1AF 8314B8F3 7261186A
225370F9 A62E13EB F77157EB 470919C5 173D744B F24DE198 D3B10C06 FDA3C3ED

difference entering the second application. These intermediate differences can be corrected by the same message difference that we use in the first block. Thus, by using the same message difference in the second block the difference of the intermediate value is corrected. We expect to get $g'_2 = h'_1$ (i.e., the differences in the registers after the last round of the compression function are equal to the intermediate value differences), which with probability 1/4 cancels the differences after the final addition of $h_2 = g_2 + h_1$.

5 Strength of Reduced Versions of SHA-1 with More Rounds

SHA-1 with more than 40 rounds is also vulnerable to the attacks described in this paper. Though all the disturbance vectors that we found have consecutive disturbances in the first 17 rounds, many of them contain correctable consecutive disturbances. We therefore list here two set of results: the first is the results for SHA-1 reduced to fewer rounds, where these rounds are set at the first rounds of SHA-1, i.e., the first 20 rounds use the IF function. This case is denoted later by SHA-1. The second set of results, denoted later by NO-IF, have consecutive disturbances, so if the reduced version starts with 20 IF rounds, the probability of success is reduced to 0, but if the reduced version of SHA-1 starts at a different location, the attack is still possible (such as when the reduced version contains the last rounds of SHA-1, rather than the first ones).

Table 11 lists the results for 34 up to 61 rounds. For each number of rounds, and each set of results (SHA-1 or NO-IF) the table lists the Hamming weight of the disturbance vector from rounds 20 and on for three cases: the first, marked by HW, is the Hamming weight of the best disturbance vector predicting a one-block collision we found. The second, marked by 2B, is the best disturbance vector predicting a two-block collision, and the last, marked by NC is the best disturbance vector predicting a near-collision. Entries that we used to actually find a collision are marked in boldface.

Table 11. The Hamming Weights of the Best Disturbance Vectors that We Found (Counted from Round 20)

Rounds	SHA-1			NO-IF		
	HW	2B	NC	HW	2B	NC
34	**2**			2		
35	7	6	3	**4**	5	3
36	**7**	**3**	3	5	3	3
37	11	9	3	**5**	5	3
38	12	7	4	8	6	3
39	12	11	5	8	8	4
40	19	**5**	5	11	5	5
41	17	14	6	12	10	6
42	17	14	7	13	11	7
43	17	15	8	17	13	7
44	19	17	9	15	15	8
45	25	16	10	15	15	10
46	25	18	10	23	13	10
47	26	23	12	24	21	11

Rounds	SHA-1			NO-IF		
	HW	2B	NC	HW	2B	NC
48	28	25	13	14	14	13
49	32	22	15	14	14	14
50	35	29	16	14	14	14
51	38	26	19	15	15	15
52	42	32	19	16	16	15
53	42	32	20	16	16	16
54	39	42	24	36	34	16
55	39	48	27	39	38	16
56	41	39	28	41	29	16
57	61	56	29	42	23	17
58	58	52	29	42	17	17
59	64	53	29	51		17
60	45	45		29		18
61	45	38		30		19

The complexities of the attacks that use the mentioned disturbance vectors can be approximated by $2^{3\mathrm{HW}}$, where HW is the Hamming weight of the disturbance vector from round 20 and on (i.e., the value in the table). The exact complexity may vary (between 2^{2HW} to 2^{4HW}) by some factor which depends on the exact functions (IF, MAJ, XOR) used, by the rounds where the disturbances occur, and by a few additional details.

We can thus see that entries with up to about 26 Hamming weight predict a collision with complexity (slightly) faster than the generic birthday attack (as $2^{3\cdot 26} = 2^{78} < 2^{80}$). We marked the location of this threshold by underlines. Hamming weights much smaller than 26 predict much more practical complexities, and as can be seen from the table, Hamming weights up to about 10 require only a short computation on a personal computer (all the found collisions marked in boldface were found within a few seconds of computation).

It is especially interesting to see the huge increase of the Hamming weight in the case of NO-IF after 53 rounds, where the Hamming weight of 53 rounds is 16 and of 54 rounds is 36. Similarly in the two-block attack the Hamming weight is 17 for 58 rounds. Thus, we expect that one-block collisions of 53-round reduced SHA-1 can be found with complexity about 2^{60}, and two-block collisions of 58-round SHA-1 can be found with complexity about 2^{75} (this is a more accurate approximation than 2^{3HW} for this case), where the reduction is to the last 53 (respectively 58) rounds of SHA-1, but we have no hope according to the table to find one-block collisions of 54-round reductions. In the case of the first rounds of SHA-1, the maximal number of rounds according to the table is 51 using the two-block technique, but the complexity of this attack would be only marginally faster than the birthday attack (though much easier to parallelize).

We are now working on improvements for further rounds, some of them are by applications of the techniques described in this paper in more complex ways, and some using new ideas. Note that the NC column is a lower bound for any multiple-block attack, thus we see that there is still some hope for the attacker to find better results.

In particular, we succeed to show that the NO-IF figures hold also for the case of the first rounds of SHA-1 (starting with IF rounds) by using different characteristic paths for the first rounds, but leaving the same input, output, and message differences.

6 Summary

This paper presents various attacks on reduced versions of SHA-0 and SHA-1 along with various techniques for the analysis of hash functions. These techniques, along with the neutral bit technique and other prior techniques, form a set of tools that enable practical attacks on the full SHA-0, and reduces the complexity of attacking SHA-1 reduced to 58 or fewer rounds to less than the complexity of the birthday attack.

As this work is still in progress, we expect to further improve some of the attacks presented in this paper, and to incorporate several new ideas that may increase the total number of rounds that we can attack, such as three-block attacks and attacks with more than three blocks. In particular, it is possible to use the 53-round and the 58-round attacks on SHA-1 even against the first 53 and 58 rounds.

Finally we observe that a search for one-block near-collisions is easier than search for one-block collisions, as when searching for near-collisions, there is no need to fix the initial value of the compression function, but instead it is possible to fix an intermediate value, and search backwards in the direction of the initial value, and then forward for the output. In such a search, we found that the number of neutral bits is much larger than in the regular case, thus allowing to increase the number of rounds that we get for free from about 20–22 rounds to about 30 rounds, thus decreasing the number of rounds that should be analyzed by the probabilistic stage. Moreover, it is possible to select the 30 rounds to be the 30 consecutive rounds with the lowest probability in the characteristic, thus increasing the probability even further. For example, with such a technique it is possible to find pseudo-collisions of the full SHA-0 with probability about 2^{30}–2^{33}.

References

1. Eli Biham, Rafi Chen, *Near-Collisions of SHA-0*, Advances in Cryptology, proceedings of CRYPTO 2004, LNCS 3152, pp. 290–305, Springer Verlag, 2004.
2. Florent Chabaud, Antoine Joux, *Differential Collisions in SHA-0*, Advanced in Cryptology, proceedings of CRYPTO '98, LNCS 1462, pp. 56–71, Springer Verlag, 1999.

3. National Institute of Standards and Technologies, *Secure Hash Standard*, Federal Information Processing Standards Publication, FIPS-180, May 1993.
4. National Institute of Standards and Technologies, *Secure Hash Standard*, Federal Information Processing Standards, Publication FIPS-180-1, April 1995.
5. Ron Rivest, *The MD4 Message-Digest Algorithm*, Network Working Group, Request for Comments:1186, October 1990.
6. Ron Rivest, *The MD5 Message-Digest Algorithm*, Network Working Group, Request for Comments:1321, April 1992.
7. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu, *Cryptanalysis for Hash Functions MD4 and RIPEMD*, these proceedings.
8. Xiaoyun Wang, Hongbo Yu, *How to Break MD5 and Other Hash Functions*, these proceedings.
9. V. Rijmen, E. Oswald. Update on SHA-1. In *RSA Crypto Track 2005*, LNCS 3376, 2005.

Reducing Complexity Assumptions for Statistically-Hiding Commitment

Iftach Haitner[1,⋆], Omer Horvitz[2,⋆⋆], Jonathan Katz[2,⋆⋆⋆], Chiu-Yuen Koo[2], Ruggero Morselli[2], and Ronen Shaltiel[3]

[1] Department of Computer Science, Weizmann Institute of Science
iftach.haitner@weizmann.ac.il
[2] Department of Computer Science, University of Maryland
{horvitz, jkatz, cykoo, ruggero}@cs.umd.edu
[3] Department of Computer Science, University of Haifa
ronen@cs.haifa.ac.il

Abstract. Determining the minimal assumptions needed to construct various cryptographic building blocks has been a focal point of research in theoretical cryptography. Here, we revisit the following question: *what are the minimal assumptions needed to construct statistically-hiding commitment schemes*? Previously, it was known how to construct such schemes based on one-way permutations. We improve upon this by constructing statistically-hiding commitment schemes based on *approximable-preimage-size* one-way functions. These are one-way functions for which there is an efficient way to approximate the number of preimages of a given output. A special case (for which we show a somewhat simpler construction) is that of *regular* one-way functions where all outputs have the same number of preimages.

We utilize two different approaches in constructing statistically-hiding commitment schemes. Our first approach proceeds by showing that the scheme of Naor et al. can be implemented using any one-way function having an output distribution which is "sufficiently similar" to uniform. We then construct one-way functions with this property from approximable-preimage-size one-way functions. Our second approach begins by constructing a commitment scheme which is statistically hiding against an honest-but-curious receiver. We then demonstrate a *compiler* which transforms any such commitment scheme into one which is statistically hiding even against a malicious receiver. This compiler and its analysis may be of independent interest.

1 Introduction

A central focus of modern cryptography has been to investigate the weakest possible assumptions under which various cryptographic primitives exist. This

⋆ Research supported by U.S.-Israel Binational Science Foundation grant 2002246.
⋆⋆ Research supported by U.S. Army Research Office award DAAD19-01-1-0494.
⋆⋆⋆ Supported by NSF CAREER award 0447075.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 58–77, 2005.

direction of research has been quite fruitful, and minimal assumptions are known for a wide variety of primitives: e.g., pseudorandom generators, pseudorandom functions, symmetric-key encryption/message authentication, and digital signatures [21, 12, 13, 20, 24, 26, 29]. In other cases, black-box separation results exist which indicate the difficulty — if not impossibility — of constructing "strong" cryptographic protocols (say, key-exchange) from "weak" building blocks (say, one-way permutations; see [22]).

The above may give the impression that exact characterizations for all primitives of interest (at least in terms of equivalent complexity-theoretic assumptions) are known; however, this is not the case. Questions that remain open (to choose two examples) include the possibility of constructing efficient-prover non-interactive zero-knowledge proofs [4] based on assumptions weaker than trapdoor permutations [9], as well as determining whether constant-round ZK proofs exist based only on the assumption of one-way functions (see [10–Chap. 4]).

Another key cryptographic primitive in which a weakest possible assumption is not known is *statistically-hiding commitment*. Informally, a commitment scheme defines a two-phase interactive protocol between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$; after the *commitment phase*, $\mathcal{S}$ is uniquely bound to (at most) one value which is not yet revealed to $\mathcal{R}$, and in the *decommitment phase* $\mathcal{R}$ finally learns this value. The two security properties hinted at in this informal description are known as *binding* (namely, that $\mathcal{S}$ is bound to at most one value after the commitment phase) and *hiding* (namely, that $\mathcal{R}$ does not learn the value to which $\mathcal{S}$ commits before the decommitment phase). In a statistically-hiding commitment scheme the hiding property holds *even against all-powerful receivers* (i.e., hiding holds information-theoretically), while the binding property is required to hold only for computationally-bounded (say, polynomial-time) senders.

Statistically-hiding commitment schemes can be used as a building block in constructions of statistical zero-knowledge arguments [6, 25] or certain coin-tossing protocols [2, 23]. They are also advantageous when used within protocols in which certain commitments are never revealed; in this case, it need only be infeasible to violate the binding property *during the period of time the protocol is run*, whereas the committed values will remain hidden *forever* (i.e., regardless of how much time the receiver invests after completion of the protocol). Indeed, this is part of the motivation for statistical zero-knowledge as well. For further discussion, the reader is referred to [27, 28, 25].

Perfectly-hiding[1] commitment schemes were first shown to exist based on specific number-theoretic assumptions [6, 5] or, more generally, based on any collection of claw-free permutations [18] with an efficiently-recognizable index set [15] (see [15] for a definition of a weaker variant of statistically-hiding commitment which suffices for some applications and for which an efficiently-recognizable

[1] Very informally, in a statistically-hiding commitment scheme the receiver learns only a negligible amount of information about the sender's committed value, whereas in a perfectly-hiding commitment scheme the receiver learns *nothing*. Note that any perfectly-hiding scheme is also statistically-hiding.

index set is not needed). Naor, et al. [25], using techniques developed earlier by Ostrovsky, et al. [27, 28], later showed a construction of a perfectly-hiding commitment scheme based on one-way permutations. Statistically-hiding commitment schemes can also be constructed from collision-resistant hash functions [8, 19] (see [30] for minimal assumptions for the existence of the latter).

1.1 Our Results

We show how to construct a statistically-hiding commitment scheme given any *approximable-preimage-size* one-way function. Informally, this is a one-way function f satisfying the additional property that, given any y in the image of f, the value $|\{x : f(x) = y\}|$ (i.e., the number of points mapping to y) can be efficiently estimated. An interesting special case, for which our construction may be somewhat simplified, is that of *regular* one-way functions for which every point in the image of f has the same number of preimages. (We still require that it be feasible to approximate the number of preimages.) A variety of conjectured one-way functions are regular; we refer the reader to [16] for examples.

We show two different approaches to constructing statistically-hiding commitment schemes: the first is more direct and achieves better computational efficiency, while the second achieves better round complexity (in fact, it achieves round complexity identical to [25]). As part of our second approach, we show a *compiler* transforming any commitment scheme which is statistically-hiding against an honest-but-curious (a.k.a. semi-honest) receiver into one which is statistically-hiding against an arbitrarily-malicious receiver. Since our compiler requires only the existence of one-way functions, our result implies an equivalence between the two formulations of the problem. (Due to space limitations the details of our second approach do not appear in this version.)

Our results may be viewed as an example of the paradigm in which a sequence of works constructs a given primitive from ever-weaker assumptions; e.g., in the cases of pseudorandom generators and universal one-way hash functions/signature schemes (see [10–Chap. 2] and [11–Chap. 6]), constructions were first based on specific, number-theoretic assumptions [3, 18], and then the minimal assumptions were gradually reduced to trapdoor permutations [1] (in the case of signatures), one-way permutations [17, 26], regular one-way functions [16, 31], and (finally) one-way functions [20, 29]. We hope our work will similarly serve as a step toward resolving the question of the minimal assumptions required for statistically-hiding commitment.

1.2 Overview of Our Techniques

Our constructions are based on the protocol of Naor et al. [25], which is shown there to be perfectly hiding (as well as computationally binding) when applied using a one-way permutation. It is natural to ask what happens when this protocol is applied using some other function $f : \{0,1\}^n \to \{0,1\}^\ell$. We first observe that the main argument of [25] shows that the protocol is computationally binding as long as f cannot be efficiently inverted with respect to the uniform

distribution U_ℓ (more formally, no efficient algorithm can compute $f^{-1}(y)$, for uniformly-chosen y, with non-negligible probability). We call a function with this property *one-way over its range*. Note that a function with this property is not necessarily one-way.

As our first main technical result, we then show that the protocol of Naor et al. is "somewhat hiding" when applied using a function f for which the distribution $f(U_n)$ is *balanced*. (By "somewhat hiding" we mean that the receiver cannot guess the committed bit with probability better than some constant $\rho < 1$. Such a protocol can be "amplified" using repetition to give a statistically-hiding protocol.) Loosely speaking, a distribution over $\{0,1\}^\ell$ is balanced if it assigns to "most" elements $y \in \{0,1\}^\ell$ a probability that is close to $2^{-\ell}$ (say between $(99/100) \cdot 2^{-\ell}$ and $(101/100) \cdot 2^{-\ell}$). (In the precise definition we allow some elements to have probability outside this range as long as both the number of such elements and their total weight are small.)

The remainder of the paper is devoted to constructing functions that are both balanced and one-way over their range.[2] Intuitively, both these properties require the output distribution $f(U_n)$ to be "somewhat similar" to uniform. While we do not know how to construct such a function given a general one-way function, we show how to construct such functions given regular or approximable-preimage-size one-way functions. We achieve this goal using polywise independent hashing, inspired by [20, 29]. More precisely, given a regular one-way function f (the case of approximable-preimage-size one-way functions is more complex), we define $f'(h,x) = (h, h(f(x)))$ where h is selected from a family of $O(k)$-wise independent hash functions (here, k is the security parameter). This hashing "smoothes" the output distribution, and we show that by choosing the output length of h appropriately we obtain an f' which is both balanced and one-way over its range. Note that making the output length of h "too small" makes f' more balanced, but possibly no longer one-way over its range (and vice versa); we use the fact that f is regular (and that the number of preimages is known) when setting the output length of h. This is why our approach does not extend for general one-way functions.

Due to space limitations, some proofs have been omitted or shortened.

2 Preliminaries

Throughout this paper, we let k denote the security parameter. If X_1 and X_2 are two distributions over a set $\mathcal{X}$, their statistical difference (written $\mathsf{SD}(X_1, X_2)$) is defined as:

$$\mathsf{SD}(X_1, X_2) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{x \in \mathcal{X}} \left| \Pr_{X_1}[x] - \Pr_{X_2}[x] \right| .$$

[2] We remark that known constructions of "almost-everywhere one-to-one" one-way functions [14], "almost one-to-one" one-way functions [10–Sect. 3.5], and the constructions of [20] do not suffice for our purposes.

Two distribution ensembles $\mathcal{X}_1 = \{X_1(k)\}_{k\in\mathbb{N}}$ and $\mathcal{X}_2 = \{X_2(k)\}_{k\in\mathbb{N}}$ have statistical difference ρ (for ρ a function of k) if $\mathsf{SD}(X_1(k), X_2(k)) \leq \rho(k)$ for all k large enough. If ρ is negligible, we say the ensembles are *statistically indistinguishable*. For a function $f : \{0,1\}^n \to \{0,1\}^\ell$, we let $\mathsf{image}(f) \stackrel{\text{def}}{=} \{f(x) \mid x \in \{0,1\}^n\}$.

2.1 Commitment Schemes

An interactive bit commitment scheme is defined via a triple of PPT algorithms $(\mathcal{S}, \mathcal{R}_1, \mathcal{R}_2)$. Looking ahead, $\mathcal{S}$ and $\mathcal{R}_1$ will interact during what is called a *commitment phase*, while $\mathcal{R}_2$ will be used during the (non-interactive) *decommitment phase*. More formally:

- $\mathcal{S}$ (the *sender*) is an interactive Turing machine (ITM) which receives as initial input the security parameter 1^k and a bit b. Following its interaction, it outputs some information decom (the *decommitment*).
- $\mathcal{R}_1$ (the *receiver*) is an ITM which receives the security parameter 1^k as initial input. Following its interaction, it outputs some state information s.
- $\mathcal{R}_2$ (acting as a receiver, in the decommitment phase) is a deterministic algorithm which receives as input state information s and a decommitment decom; it outputs either a bit b or the distinguished value $\perp$.

Denote by $(\mathsf{decom} \mid s) \leftarrow \langle \mathcal{S}(1^k, b), \mathcal{R}_1(1^k)\rangle$ the experiment in which $\mathcal{S}$ and $\mathcal{R}_1$ interact (using the given inputs and uniformly random coins), and then $\mathcal{S}$ outputs decom while $\mathcal{R}_1$ outputs s. We make the following correctness requirement: for all k, all b, and every pair $(\mathsf{decom} \mid s)$ that may be output by $\langle \mathcal{S}(1^k, b), \mathcal{R}_1(1^k)\rangle$, it is the case that $\mathcal{R}_2(s, \mathsf{decom}) = b$.

The security of a commitment scheme can be defined in two complementary ways, protecting against either an all-powerful sender or an all-powerful receiver. Since we are interested in the case of statistically-hiding commitment (i.e., the latter case), we only provide the definition for this case.

Definition 1. Commitment scheme $(\mathcal{S}, \mathcal{R}_1, \mathcal{R}_2)$ is *ρ-hiding* (for ρ a function of k) if the following holds: Given a deterministic ITM $\mathcal{R}_1^*$, let $\mathsf{view}_{\langle \mathcal{S}(b), \mathcal{R}_1^*\rangle}(k)$ denote the distribution on the view of $\mathcal{R}_1^*$ when interacting with $\mathcal{S}(1^k, b)$ (this view simply consists of the sequence of messages it receives from $\mathcal{S}$), where this distribution is taken over the random coins of $\mathcal{S}$. Then we require that for any (even all-powerful) $\mathcal{R}_1^*$ the ensembles $\{\mathsf{view}_{\langle \mathcal{S}(0), \mathcal{R}_1^*\rangle}(k)\}$ and $\{\mathsf{view}_{\langle \mathcal{S}(1), \mathcal{R}_1^*\rangle}(k)\}$ have statistical difference at most ρ.

Note that in the above, considering a deterministic $\mathcal{R}_1^*$ is without loss of generality. We say a scheme is *statistically hiding* if it is ρ-hiding for negligible ρ. A 0-hiding scheme is called *perfectly hiding*.

Definition 2. Commitment scheme $(\mathcal{S}, \mathcal{R}_1, \mathcal{R}_2)$ is *computationally-binding* if the following is negligible for all PPT $\mathcal{S}^*$:

$$\Pr\left[((\mathsf{decom}, \mathsf{decom}') \mid s) \leftarrow \langle \mathcal{S}^*(1^k), \mathcal{R}_1(1^k)\rangle : \begin{array}{c} \mathcal{R}_2(s, \mathsf{decom}), \mathcal{R}_2(s, \mathsf{decom}') \in \{0,1\} \\ \bigwedge \mathcal{R}_2(s, \mathsf{decom}) \neq \mathcal{R}_2(s, \mathsf{decom}') \end{array}\right],$$

where the probability is taken over the random coins of both $\mathcal{S}^*$ and $\mathcal{R}_1$.

Given the above, we now define a statistically-secure commitment scheme:

Definition 3. Commitment scheme $(\mathcal{S}, \mathcal{R}_1, \mathcal{R}_2)$ is *ρ-secure* (resp., *statistically secure*, *perfectly secure*) if it is computationally binding and ρ-hiding (resp., statistically hiding, perfectly hiding).

2.2 One-Way Function Families and Variants

Let $n, \ell = \mathsf{poly}(k)$ be poly-time computable and let $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}_{k \in \mathbb{N}}$ be a function family. We say $\mathcal{F}$ is *one-way* if the following hold:

- (**efficiently computable**) There exists a (deterministic) polynomial-time algorithm E such that, for all k and all $x \in \{0,1\}^{n(k)}$, $E(1^k, x) = f_k(x)$.
- (**one-way**) For all PPT algorithms A, the following is negligible (in k):

$$\Pr_{x \leftarrow \{0,1\}^{n(k)}}[f_k(A(1^k, f_k(x))) = f_k(x)].$$

We consider two additional properties of function families:

- $\mathcal{F}$ is $r(k)$**-regular** if for every k and every $x \in \{0,1\}^{n(k)}$ we have

$$\left|\{x' \in \{0,1\}^{n(k)} \mid f_k(x') = f_k(x)\}\right| = 2^{r(k)}$$

  and $r(k)$ is poly-time computable.[3] In other words, for each $x \in \{0,1\}^{n(k)}$ there are exactly $2^{r(k)}$ elements (including x itself) which f_k maps to the same value.
- $\mathcal{F}$ is **approximable-preimage-size** if the function $\tilde{D}_{\mathcal{F}}(y,k) \stackrel{\text{def}}{=} \lceil \log(|f_k^{-1}(y)|\,) \rceil$ is polynomial-time computable.[4]

For simplicity, we drop the explicit dependence on k when clear. Note that any regular function family is also approximable-preimage-size.

2.3 Entropy Measures

Let U_n denote the uniform distribution over $\{0,1\}^n$. Given a function $f : \{0,1\}^n \to \{0,1\}^\ell$, we let $f(U_n)$ denote the distribution over $\{0,1\}^\ell$ induced by f operating on the uniform distribution. Given a distribution D over some set X, the *support* of D is defined to be the set $\{x \in X | D(x) > 0\}$. For D a distribution over some finite domain X, we use the following "measures" of entropy:

- The *min-entropy* of D is $H_\infty(D) \stackrel{\text{def}}{=} \min_{x \in X} \log(\frac{1}{D(x)})$.
- The *max-entropy* of D is $H_{max}(D) \stackrel{\text{def}}{=} \max_{x \in X} \log(\frac{1}{D(x)})$.

[3] Some previous definitions of regular functions do not require that r be poly-time computable. However, we do not know how to extend our results to this case.

[4] Our constructions generalize to the case where $r(k)$ (resp., $\tilde{D}_{\mathcal{F}}(y,k)$) are not computed precisely, but rather approximated to within an additive factor of $O(\log(k))$.

– The *Renyi entropy* of D is $H_2(D) \stackrel{\text{def}}{=} \log(\frac{1}{CP(D)})$, where $CP(D) \stackrel{\text{def}}{=} \sum_{x \in X} D(x)^2$ is the *collision probability* of D.

We will be interested in distributions of the form $D = f(U_n)$ for $f : \{0,1\}^n \to \{0,1\}^l$. Note that if f is r-regular, then D is uniform over some subset of $\{0,1\}^\ell$ and the above three measures coincide (and D has entropy $t = n - r$).

2.4 Universal Hashing and an Extended Chernoff Bound

Let $\mathcal{H} = \{H_k\}_{k \in \mathbb{N}}$ be a sequence of function families, where each H_k is a family of functions mapping strings of length $\ell(k)$ to strings of length $v(k)$. We say H_k is an $n(k)$-*universal hash family* (following [7]) if for any distinct $x_1, \ldots, x_{n(k)} \in \{0,1\}^{\ell(k)}$, and any $y_1, \ldots, y_{n(k)} \in \{0,1\}^{v(k)}$ we have:

$$\Pr_{h \leftarrow H_k}[h(x_1) = y_1 \wedge \cdots \wedge h(x_n) = y_n] = 2^{-v(k) \cdot n}.$$

In this paper, it is convenient to assume that for every k, the size of H_k is a power of two. This allows us to identify functions $h \in H_k$ with binary strings. We use $s(k)$ to denote the length of these strings.

We say that $\mathcal{H}$ is an $n(k)$-universal hash family if for every k, H_k is an $n(k)$-universal hash family and furthermore there is a polynomial time algorithm that given 1^k, $x \in \{0,1\}^{n(k)}$ and a string $h \in \{0,1\}^{s(k)}$ outputs $h(x)$ (where $h \in H_k$ is the function described by the string $h \in \{0,1\}^{s(k)}$). It is well-known that there is a family of functions with this property for every choice of ℓ and v with $s(k) = O(n(k) \cdot \max(\ell(k), v(k)))$.

The following Chernoff-like bound will be useful in our analysis:

Lemma 1. (Extended Chernoff Bound [32–Theorem 5]) *Let X be the sum of (any number of) n-wise independent random variables, each taking values in the interval $[0,1]$, such that $E[X] = \mu$. Then for any $\varepsilon \leq 1$ for which $n \geq \lfloor \varepsilon^2 \mu e^{-1/3} \rfloor$ we have $\Pr[|X - \mu| \geq \varepsilon\mu] \leq e^{-\lfloor \varepsilon^2 \mu/3 \rfloor}$.*

2.5 Interactive Hashing and the Construction of [25]

Interactive hashing was introduced by Ostrovsky, et al. [27, 28], and used by Naor, et al. [25] to construct a statistically-secure (actually, perfectly-secure) commitment scheme based on any one-way permutation family. We review interactive hashing, as well as the resulting commitment scheme, below. In what follows, we let $x \cdot y$ denote $\sum_{i=1}^m x_i y_i \bmod 2$ for $x, y \in \{0,1\}^m$.

Construction 4 (Interactive hashing). *The protocol is defined by algorithms $\mathcal{S}$ and $\mathcal{R}$, where $\mathcal{S}$ begins with an m-bit value y (with m known to $\mathcal{R}$), and proceeds as follows:*

1. *The parties interact in $m - 1$ stages. In stage i (for $i = 1, \ldots, m-1$), $\mathcal{R}$ chooses $r_i \in \{0,1\}^{m-i}$ uniformly at random and sends the "query" $q_i = 0^{i-1}1r_i$ to $\mathcal{S}$ (in case $\mathcal{R}$ aborts, $\mathcal{S}$ simply takes q_i to be some default value); in response, $\mathcal{S}$ sends $c_i = q_i \cdot y$.*

2. *At the conclusion of the above, there are exactly two strings $y_0, y_1 \in \{0,1\}^m$ satisfying the system of equations $\{q_i \cdot X = c_i\}_{1 \le i \le m-1}$; let y_0 denote the lexicographically smaller of the two. Both parties compute (y_0, y_1), and $\mathcal{S}$ chooses v such that $y = y_v$.*

We define the output of the protocol to be (y_0, y_1, v) for $\mathcal{S}$ and (y_0, y_1) for $\mathcal{R}$. We denote by $IH(y)$ an execution of the interactive hashing protocol, where $\mathcal{S}$ begins with input y.

The above was used in [25] to construct a perfectly-secure commitment scheme based on one-way permutations via the following approach:

Construction 5. *Let $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}$ be a function family. Commitment scheme $(\mathcal{S}, \mathcal{R}_1, \mathcal{R}_2)$ is defined as follows: $\mathcal{S}(1^k, b)$ chooses $x \in \{0,1\}^{n(k)}$ uniformly at random, computes $y = f_k(x)$, and then executes $IH(y)$ with $\mathcal{R}_1$; this protocol results in output (y_0, y_1, v) for $\mathcal{S}$ and (y_0, y_1) for $\mathcal{R}_1$. The commitment phase concludes by having $\mathcal{S}$ send $\hat{v} = v \oplus b$ to $\mathcal{R}_1$. Finally, $\mathcal{S}$ outputs $\mathsf{decom} = x$ while $\mathcal{R}_1$ outputs state $s = (y_0, y_1, \hat{v})$.*

In the decommitment phase, $\mathcal{R}_2((y_0, y_1, \hat{v}), x)$ proceeds as follows: if $f_k(x) = y_0$, output $\hat{v}$; if $f_k(x) = y_1$, output $\hat{v} \oplus 1$; otherwise, output $\perp$.

It is relatively easy to observe that the above protocol is perfectly hiding if $\mathcal{F}$ is a permutation family (regardless of whether $\mathcal{F}$ is one-way). The main result of [25] was to prove that the above is *computationally binding* when $\mathcal{F}$ is a *one-way* permutation family. In fact, careful examination of their proof shows the above commitment scheme is computationally binding under a *weaker* condition on $\mathcal{F}$; it suffices for $\mathcal{F}$ to be what we call "one-way over its range", defined as follows:

Definition 6. Let $n, \ell = \mathsf{poly}(k)$ be poly-time computable functions and let $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}_{k \in \mathbb{N}}$ be a function family. We say $\mathcal{F}$ is *one-way over its range* if the following hold:

- (**efficiently computable**) There exists a (deterministic) polynomial-time algorithm E such that, for all k and all $x \in \{0,1\}^{n(k)}$, $E(1^k, x) = f_k(x)$.
- (**one-way over range**) For all PPT A, the following is negligible (in k):

$$\Pr_{y \leftarrow \{0,1\}^{\ell(k)}}[f_k(A(1^k, y)) = y].$$

Theorem 1 (Implicit in [25]). *If $\mathcal{F}$ is one-way over its range, then Construction 5 is computationally binding.*

3 Statistical Hiding from Balanced Functions

In this section we define a notion of "balance" and show that if a function family $\mathcal{F}$ is "sufficiently balanced" then Construction 5 yields a protocol that is "somewhat hiding". Roughly speaking, a distribution D on $\{0,1\}^\ell$ is balanced if D is "close" to uniform "most" of the time. A function $f : \{0,1\}^n \to \{0,1\}^\ell$ is then defined to be balanced if the distribution $f(U_n)$ is balanced. Formally:

Definition 7. Distribution D on $\{0,1\}^\ell$ is (α,δ)-*balanced* if there is a set $\mathsf{Bad} \subset \{0,1\}^\ell$ such that:

1. $|\mathsf{Bad}| \leq \alpha \cdot 2^\ell$.
2. $\Pr_{y\leftarrow D}[y \in \mathsf{Bad}] \leq \alpha$.
3. For every $y_0 \notin \mathsf{Bad}$, $\left|\Pr_{y\leftarrow D}[y = y_0] - \frac{1}{2^\ell}\right| \leq \frac{\delta}{2^\ell}$ (we will always have $\delta < 1$).

Function $f : \{0,1\}^n \to \{0,1\}^\ell$ is (α,δ)-*balanced* if the distribution $f(U_n)$ is (α,δ)-balanced. Function family $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}$ is (α,δ)-*balanced* if, for all k large enough, f_k is $(\alpha(k),\delta(k))$-balanced.

Our main result of this section is the following:

Theorem 2. *If $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}$ is an (α,δ)-balanced function family, then Construction 5 is ρ-hiding for $\rho = 2\alpha + \delta + \alpha\delta$.*

Proof. Fix k large enough so that f_k is $(\alpha(k),\delta(k))$-balanced; from now on we simply write f, α, δ, ρ without explicitly indicating their dependence on k. For a given execution of the scheme, let τ denote the initial transcript resulting from the interactive hashing sub-protocol; thus, the view of $\mathcal{R}_1^*$ consists of τ and the bit $\hat{v}$ sent in the final round. Given a particular (deterministic) $\mathcal{R}_1^*$, we therefore write $\mathsf{Exp(b)} \stackrel{\text{def}}{=} (\tau,\hat{v}) \leftarrow \mathsf{view}_{\langle \mathcal{S}(b),\mathcal{R}_1^*\rangle}$ (cf. Definition 3) to denote the experiment in which $\mathcal{S}$ chooses a uniform random tape and then executes the protocol with $\mathcal{R}_1^*$ using this random tape and the bit b, resulting in view $(\tau,\hat{v})$ for $\mathcal{R}_1^*$. Below, we define a "good" set of initial transcripts Good, and show that:

Claim. With probability at least $1 - \alpha(2+\delta)$, we have $\tau \in \mathsf{Good}$.

Claim. The following holds for all $\tau^* \in \mathsf{Good}$ and $\hat{v}^* \in \{0,1\}$:

$$\left| \Pr_{\mathsf{Exp(0)}}[\hat{v} = \hat{v}^* \mid \tau = \tau^*] - \Pr_{\mathsf{Exp(1)}}[\hat{v} = \hat{v}^* \mid \tau = \tau^*] \right| \leq \delta.$$

These claims suffice to prove the Theorem, since the statistical difference between the view of $\mathcal{R}_1^*$ when the sender commits to 0 (i.e., $b = 0$) and the view of $\mathcal{R}_1^*$ when the sender commits to 1 (i.e., $b = 1$) may be bounded as follows:

$$\begin{aligned}
&\frac{1}{2}\sum_{\tau^*,\hat{v}^*} \left| \Pr_{\mathsf{Exp(0)}}[(\tau,\hat{v}) = (\tau^*,\hat{v}^*)] - \Pr_{\mathsf{Exp(1)}}[(\tau,\hat{v}) = (\tau^*,\hat{v}^*)] \right| \\
&= \frac{1}{2}\sum_{\tau^*,\hat{v}^*} \left| \Pr_{\mathsf{Exp(0)}}[\tau=\tau^*] \Pr_{\mathsf{Exp(0)}}[\hat{v}=\hat{v}^*|\tau=\tau^*] - \Pr_{\mathsf{Exp(1)}}[\tau=\tau^*] \Pr_{\mathsf{Exp(1)}}[\hat{v}=\hat{v}^*|\tau=\tau^*] \right| \\
&\leq \Pr[\tau \notin \mathsf{Good}] + \frac{1}{2}\sum_{\tau^*\in\mathsf{Good},\hat{v}^*} \Pr[\tau=\tau^*] \left| \Pr_{\mathsf{Exp(0)}}[\hat{v}=\hat{v}^*|\tau=\tau^*] - \Pr_{\mathsf{Exp(1)}}[\hat{v}=\hat{v}^*|\tau=\tau^*] \right| \\
&\leq \alpha(2+\delta) + \frac{1}{2}\sum_{\tau^*\in\mathsf{Good};\hat{v}^*} \Pr[\tau=\tau^*]\cdot\delta \quad \leq \quad \alpha(2+\delta) + \delta,
\end{aligned}$$

where we use the fact that $\Pr_{\mathsf{Exp}(0)}[\tau = \tau^*] = \Pr_{\mathsf{Exp}(1)}[\tau = \tau^*]$ for any τ^*, since the initial transcript τ does not depend on b.

We proceed with the proof of the first claim by defining the set of good initial transcripts. Let $\mathsf{Bad} \subset \{0,1\}^\ell$ be the subset whose existence is guaranteed by Definition 7 (using the fact that f is balanced). Recall that the initial transcript τ defines two strings $y_0^\tau, y_1^\tau \in \{0,1\}^\ell$ (cf. Construction 4). We say $\tau \in \mathsf{Good}$ iff $y_0^\tau, y_1^\tau \notin \mathsf{Bad}$.

We first bound the probability that $y_v = y$ is in Bad (we are using here the notation from Construction 5). Since f is (α, δ)-balanced and since the value of y depends only on the choices of the sender (who is assumed honest here), it follows that this probability is at most α.

Next, we bound the probability that $y_v \notin \mathsf{Bad}$ but $y_{\bar{v}} \in \mathsf{Bad}$. Since f is balanced, we have $|\mathsf{Bad}| \leq \alpha 2^\ell$. Now, since $\mathcal{R}_1^*$ is deterministic, we have that $y_{\bar{v}}$ is uniquely determined by y_v. Let ϕ be the function mapping the sender's chosen value y_v to the second value $y_{\bar{v}}$ resulting from the interactive hashing protocol. Observe that if $\phi(y) = y'$ then $\phi(y') = y$; this is because, for either of these choices, the sender responds with the exact same answer to each of the receiver's queries during the interactive hashing sub-protocol. It follows that ϕ is a permutation. Letting $\mathsf{MapToBad} \stackrel{\text{def}}{=} \phi^{-1}(\mathsf{Bad})$, we get:

$$\begin{aligned}
\Pr\left[y_v \notin \mathsf{Bad} \bigwedge y_{\bar{v}} \in \mathsf{Bad}\right] &= \Pr\left[y_v \in \mathsf{MapToBad} \setminus \mathsf{Bad}\right] \\
&= \sum_{y^* \in \mathsf{MapToBad} \setminus \mathsf{Bad}} \Pr\left[y_v = y^*\right] \\
&\leq \sum_{y^* \in \mathsf{MapToBad} \setminus \mathsf{Bad}} (1+\delta)\frac{1}{2^\ell}
\end{aligned}$$

using the definition of Bad. Continuing:

$$\begin{aligned}
\sum_{y^* \in \mathsf{MapToBad} \setminus \mathsf{Bad}} (1+\delta)\frac{1}{2^\ell} &= |\mathsf{MapToBad} \setminus \mathsf{Bad}| \cdot (1+\delta)\frac{1}{2^\ell} \\
&\leq |\mathsf{MapToBad}| \cdot (1+\delta)\frac{1}{2^\ell} \\
&\leq (1+\delta) \cdot \alpha \qquad (1)
\end{aligned}$$

(using the fact that $|\mathsf{MapToBad}| = |\mathsf{Bad}|$). It follows that $\tau \notin \mathsf{Good}$ with probability at most $(2+\delta) \cdot \alpha$, completing the proof of the first claim.

We proceed to prove the second claim. Let $P(\tilde{y}) \stackrel{\text{def}}{=} \Pr_{x \in \{0,1\}^n}[f(x) = \tilde{y}]$. For any τ^* and any $\hat{v}^* \in \{0,1\}$ we have

$$\begin{aligned}
\Pr_{\mathsf{Exp(b)}}[\hat{v} = \hat{v}^* \mid \tau = \tau^*] &= \Pr_{\mathsf{Exp(b)}}[v = \hat{v}^* \oplus b \mid \tau = \tau^*] \\
&= \Pr_{\mathsf{Exp(b)}}[y = y^{\tau^*}_{\hat{v}^* \oplus b} \mid y \in \{y_0^{\tau^*}, y_1^{\tau^*}\}] \\
&= \frac{P(y^{\tau^*}_{\hat{v}^* \oplus b})}{P(y_0^{\tau^*}) + P(y_1^{\tau^*})}.
\end{aligned}$$

If $\tau^* \in \mathsf{Good}$, then $y_0^{\tau^*}, y_1^{\tau^*} \notin \mathsf{Bad}$ and so $P(y_0^{\tau^*}), P(y_1^{\tau^*})$ lie in the range $[(1-\delta)2^{-\ell}, (1+\delta)2^{-\ell}]$. It follows that when $\tau^* \in \mathsf{Good}$ the following holds for any $\hat{v}^* \in \{0,1\}$:

$$\left| \Pr_{\mathsf{Exp}(0)}[\hat{v} = \hat{v}^* \mid \tau = \tau^*] - \Pr_{\mathsf{Exp}(1)}[\hat{v} = \hat{v}^* \mid \tau = \tau^*] \right| = \frac{\left|P(y_0^{\tau^*}) - P(y_1^{\tau^*})\right|}{P(y_0^{\tau^*}) + P(y_1^{\tau^*})} \leq \delta,$$

which proves the claim. This completes the proof of the Theorem 2.

4 Achieving Our Main Result: A Roadmap

We now outline our approach to constructing statistically-secure commitment schemes based on assumptions weaker than one-way permutations. It follows from Theorems 1 and 2 that if we can construct an (α, δ)-balanced $\mathcal{F}$ that is also one-way over its range, then we can construct a ρ-secure commitment scheme for $\rho = O(\alpha + \delta)$. For α and δ sufficiently-small constants we thus obtain a ρ-secure commitment scheme for some constant $\rho < 1$. Using standard techniques, we can then "amplify" this scheme to obtain a statistically-secure commitment scheme. (Exact details of this amplification will appear in the full version.)

It remains to construct $\mathcal{F}$ with the desired properties. In Section 5 we show how to construct such an $\mathcal{F}$ based on any regular one-way function family, while in Section 6 we show how to base the construction on an approximable-preimage-size one-way function family. These, in turn, yield statistically-secure commitment schemes based on these assumptions. Altogether we conclude that:

Theorem 3 (Main Theorem). *If there exists an approximable-preimage-size one-way function family then there exists a statistically-secure commitment scheme.*

5 Starting from Regular One-Way Functions

In this section we show a construction of statistically-secure commitment based on any regular one-way function family. More concretely, given an $r(k)$-regular one-way function family $\mathcal{F}$, we show how to construct a balanced function $\mathcal{F}'$ which is also one-way over its range. Note that $n(k) - r(k)$ measures the entropy of the output distribution of f_k, and this holds for all the measures of entropy defined in this paper.

Construction 8. *Let $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}_{k\in\mathbb{N}}$ be a family of functions, let $t = t(k)$ be a function, and let $c > 0$ be a constant. Let $\mathcal{H} = \{H_k\}$ be a $3k$-universal collection of hash families where each H_k is a family of functions mapping strings of length $\ell(k)$ to strings of length $t(k) - \log(ck)$, and furthermore $|H_k| = 2^{s(k)}$ where $s(k) = \mathsf{poly}(k)$. Define:*

$$\mathcal{F}' = \left\{ f'_k : H_k \times \{0,1\}^{n(k)} \to H_k \times \{0,1\}^{t(k)-\log(ck)} \right\}_{k\in\mathbb{N}}$$

such that $f'_k(h, x) = (h, h(f_k(x)))$.

The main result of this section is the following.

Theorem 4. *Let $0 < \delta < 1$ be an arbitrary constant. Let $\mathcal{F}$ be an $r(k)$-regular one-way function family. Set $t(k) = n(k) - r(k)$, $c = 6\ln 2/\delta^2$, and let $\mathcal{F}'$ be the function family defined in Construction 8. Then $\mathcal{F}'$ is a $(2^{-k}, \delta)$-balanced and one-way over its range.*

5.1 Showing that $\mathcal{F}'$ is Balanced

We begin by showing that $\mathcal{F}'$ is $(2^{-k}, \delta)$-balanced. Preparing for the case of approximable-preimage-size one-way function families, we prove a more general statement here.

Lemma 2. *Let $c > 6\ln 2$ be an arbitrary constant and $k \geq 2$ be an integer, and set $\delta = (6\ln 2/c)^{1/2}$ and $t > \log(ck)$. Let H be a $3k$-universal hash family mapping strings of length ℓ to strings of length $t - \log(ck)$, and let Z be a distribution on $\{0,1\}^\ell$ with $H_\infty(Z) \geq t$. Then the distribution $D = \{(h, h(z))\}_{h \leftarrow H, z \leftarrow Z}$ is $(2^{-k}, \delta)$-balanced.*

Note that it follows that $\mathcal{F}'$ is $(2^{-k}, \delta)$-balanced, as the output distribution of f_k has min-entropy at least $t(k)$ (in fact, exactly $t(k)$).

Proof. For any $z \in \{0,1\}^\ell$ and $y \in \{0,1\}^{t-\log(ck)}$, define the random variable $X_{z,y}$ (over choice of $h \in H$) to take the value $2^t \cdot \Pr_Z[z]$ if $h(z) = y$, and 0 otherwise. Note that $X_{z,y} \in [0,1]$ since Z has min-entropy at least t. Let $X_y \stackrel{\text{def}}{=} \sum_{z \in \{0,1\}^\ell} X_{z,y}$. For any z, y we have $E[X_{z,y}] = \Pr_{h \leftarrow H}[h(z) = y] \cdot 2^t \cdot \Pr_Z[z] = 2^{-(t-\log(ck))} \cdot 2^t \cdot \Pr_Z[z] = ck \cdot \Pr_Z[z]$. It follows that

$$\mu \stackrel{\text{def}}{=} E[X_y] = \sum_z E[X_{z,y}] = ck.$$

Furthermore, since H is a $3k$-universal hash family, the random variables $\{X_{z,y}\}$ are $3k$-wise independent. Thus, by Lemma 1, we have that (for any y)

$$\Pr_h\left[\left|X_y - ck\right| \geq \delta ck\right] \leq e^{-\lfloor \mu\delta^2/3 \rfloor} < 2^{-k} \tag{2}$$

Define $\phi(h,y) \stackrel{\text{def}}{=} 2^t \cdot \sum_{z:h(z)=y} \Pr_Z[z]$, and $\mathsf{Bad} = \{(h,y) : |\phi(h,y) - ck| > \delta ck\}$. We show that, setting $\alpha = 2^{-k}$, the set Bad satisfies the three requirements of Definition 7. (Note that the quantity 2^ℓ in the text of Definition 7 becomes $|H| \cdot 2^{t-\log(ck)}$ in the current context.) Noting that $\phi(h,y) = 2^t \Pr_{z \leftarrow Z}[h(z) = y]$, observe that

$$\begin{aligned}
|\mathsf{Bad}| &= \sum_y |H| \cdot \Pr_h[(h,y) \in \mathsf{Bad}] \\
&= \sum_y |H| \cdot \Pr_h\left[\left|2^t \cdot \Pr_{z \leftarrow Z}[h(z) = y] - ck\right| > \delta ck\right] \\
&\leq 2^{t-\log(ck)} \cdot |H| \cdot 2^{-k},
\end{aligned}$$

using Eq. (2) and the fact that, once h is chosen, $X_y = 2^t \cdot \Pr_{z \leftarrow Z}[h(z) = y]$. This proves property 1.

We move on to property 2. We proceed as above except that now, for each $\xi, z \in \{0,1\}^\ell$, we define the binary random variable $R_{z,\xi}$ to be $2^t \cdot \Pr_Z[z]$ if $h(z) = h(\xi)$, and 0 otherwise. Again, $R_{z,\xi} \in [0,1]$. Let $R_\xi \stackrel{\text{def}}{=} \sum_{z \in \{0,1\}^\ell} R_{z,\xi}$. For an arbitrary $z \in \{0,1\}^\ell \setminus \{\xi\}$ we have $E[R_{z,\xi}] = 2^{-(t - \log(ck))} \cdot 2^t \cdot \Pr_Z[z] = ck \Pr_Z[z]$; also $R_{\xi,\xi} = 2^t \Pr_Z[\xi]$ with probability 1. It follows that

$$\mu' \stackrel{\text{def}}{=} E[R_\xi] = \sum_z E[R_{z,\xi}] = ck + (2^t - ck) \Pr_Z[\xi]$$

for any ξ. Note that $ck \le \mu' \le ck + 1$. Furthermore, since H is a $3k$-universal hash family, the random variables $\{R_{z,\xi}\}$ are $(3k-1)$-wise independent. Thus, by Lemma 1 we have

$$\Pr\left[|R_\xi - \mu'| \ge \frac{3}{4}\delta\mu'\right] \le e^{-\lfloor 3\mu'\delta^2/16 \rfloor} \le 2^{-k}, \tag{3}$$

where we use the fact that $\mu' \frac{9}{16}\delta^2 e^{-1/3} \le (ck+1)\frac{9}{16}\delta^2 e^{-1/3} \le 3k - 1$ (recall $k \ge 2$). We then derive:

$$\begin{aligned}\Pr_{(h,y) \leftarrow D}[(h,y) \in \mathsf{Bad}] &= \sum_\xi \Pr_Z[\xi] \cdot \Pr_h\left[\left|\phi(h, h(\xi)) - ck\right| > \delta ck\right] \\ &\le \sum_\xi \Pr_Z[\xi] \cdot \Pr_h\left[\left|R_\xi - E[R_\xi]\right| \ge \frac{3}{4}\delta E[R_\xi]\right] \le 2^{-k},\end{aligned}$$

where the first inequality uses the stated bounds on μ' and the fact that, once h is chosen, $R_\xi = 2^t \cdot \Pr_{z \leftarrow Z}[h(z) = h(\xi)]$, while the second inequality uses Eq. (3). This gives property 2.

Property 3 holds, since for any (h_0, y_0) we have

$$\Pr_{(h,y) \leftarrow D}[(h,y) = (h_0, y_0)] = \Pr_{h \leftarrow H}[h = h_0] \cdot \sum_{z : h_0(z) = y_0} \Pr_Z[z] = \frac{\phi(h_0, y_0)}{|H| 2^t}.$$

If $(h_0, y_0) \notin \mathsf{Bad}$, this probability is in the range $(1 \pm \delta)\frac{ck}{|H|2^t}$ as needed.

5.2 Showing that $\mathcal{F}'$ is One-Way over Its Range

We now show that if the initial function family $\mathcal{F}$ is one-way, then the derived function family $\mathcal{F}'$ is one-way over its range. Preparing for the case of approximable-preimage-size one-way function families, we once more prove a more general statement here. For this purpose we define the following:

Definition 9. Distribution D has (t_{Renyi}, t_{max})-*entropy* if (1) $H_2(D) \ge t_{Renyi}$, and (2) $H_{max}(D) \le t_{max}$. Function $f : \{0,1\}^n \to \{0,1\}^\ell$ has (t_{Renyi}, t_{max})-*entropy* if the distribution $f(U_n)$ has (t_{Renyi}, t_{max})-entropy. A function family $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}$ has (t_{Renyi}, t_{max})-*entropy* if, for all k large enough, f_k has $(t_{Renyi}(k), t_{max}(k))$-entropy.

Note that if f is a member of an r-regular function family then it has (t,t)-entropy for $t = n - r$. The following lemma shows that Construction 8, when given a (t_{Renyi}, t_{max})-entropy family of one-way functions, produces a function family which is one-way over its range.

Lemma 3. *Let $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}$ be a (t_{Renyi}, t_{max})-entropy one-way function family and let $c > 0$ be a constant. Let $t(k)$ be a function and let $m \geq 0$ be a constant such that $t_{max}(k) - m\log(k) \leq t(k) \leq t_{Renyi}(k)$. Let $\mathcal{F}'$ be the result of applying Construction 8 with $\mathcal{F}$, $t(\cdot)$, and c. Then $\mathcal{F}'$ is one-way over its range.*

Note that it follows that $\mathcal{F}'$ is one-way over its range by using the aforementioned observation that the regular function family $\mathcal{F}$ has $(t(k), t(k))$-entropy. We remark that the proof uses only the fact that $\mathcal{H}$ is 2-universal.

Proof. Let $v(k) \stackrel{\text{def}}{=} t(k) - \log(ck)$. We start by proving that the Renyi-entropy of the output of $\mathcal{F}'$ is high. We then use this fact to show that $\mathcal{F}'$ is one-way (in the usual sense). Finally we derive that $\mathcal{F}'$ is one-way over its range.

Claim. $H_2(f'_k(U_{s(k)}, U_{n(k)})) \geq s(k) + v(k) - 1$.

Proof.

$$\begin{aligned}
&CP(f'_k(U_{s(k)}, U_{n(k)})) \\
&= \sum_{(h,y)\in \mathsf{image}(f_k)} \left(\Pr_{(h',x)\leftarrow (H_k \times \{0,1\}^{n(k)})} [f'_k(h',x) = (h,y)] \right)^2 \\
&= \sum_{y\in\{0,1\}^{v(k)}} \sum_{h\in H_k} \frac{1}{2^{2s(k)}} \left(\sum_{z\in h^{-1}(y)} \Pr_{x\leftarrow\{0,1\}^{n(k)}} [f_k(x) = z] \right)^2 .
\end{aligned}$$

Continuing, we have:

$$\begin{aligned}
&CP(f'_k(U_{s(k)}, U_{n(k)})) \\
&= \frac{1}{2^{2s(k)}} \sum_{y\in\{0,1\}^{v(k)}} \sum_{h\in H_k} \sum_{z\in h^{-1}(y)} \left(\Pr_{x\leftarrow\{0,1\}^{n(k)}} [z] \right)^2 \\
&+ \frac{1}{2^{2s(k)}} \sum_{y\in\{0,1\}^{v(k)}} \sum_{h\in H_k} \sum_{z_1\neq z_2\in h^{-1}(y)} \\
&\times \left(\Pr_{x\leftarrow\{0,1\}^{n(k)}} [f_k(x) = z_1] \cdot \Pr_{x\leftarrow\{0,1\}^{n(k)}} [f_k(x) = z_2] \right) \\
&= \frac{1}{2^{s(k)}} CP(f_k(U_{n(k)})) + \frac{1}{2^{2s(k)}} \sum_{y\in\{0,1\}^{v(k)}} \frac{2^{s(k)}}{2^{2v(k)}} \sum_{z_1\neq z_2\in\{0,1\}^{l(k)}} \\
&\times \left(\Pr_{x\leftarrow\{0,1\}^{n(k)}} [f_k(x) = z_1] \cdot \Pr_{x\leftarrow\{0,1\}^{n(k)}} [f_k(x) = z_2] \right) \\
&\leq \frac{1}{2^{s(k)}} (CP(f_k(U_{n(k)})) + \frac{1}{2^{v(k)}}) \leq \frac{2}{2^{s(k)+v(k)}}.
\end{aligned}$$

Therefore $H_2(f'_k(U_{s(k)}, U_{n(k)})) = -\log(CP(f'_k(U_{s(k)}, U_{n(k)}))) \geq s(k) + v(k) - 1$. We now use the above claim to prove the one-wayness of $\mathcal{F}'$.

Claim. $\mathcal{F}'$ is one-way (in the usual sense).

Proof. Let A' be a PPT adversary attempting to invert $\mathcal{F}'$ and let $\mathsf{Expt}_{A'}(k)$ denote the experiment "$h \leftarrow H_k; x \leftarrow \{0,1\}^{n(k)}; (h,y) = f'_k(h,x); (h',x') \leftarrow A'(1^k, h, y)$". Let

$$\mathsf{Adv}_{A',\mathcal{F}'}(k) \stackrel{\text{def}}{=} \Pr[\mathsf{Expt}_{A'}(k) : f'_k(h',x') = (h,y)]. \tag{4}$$

Now construct a PPT adversary A (attempting to invert $\mathcal{F}$) as follows:

$\underline{A(1^k, z)}$ // $z = f_k(x)$ for some $x \in \{0,1\}^{n(k)}$ chosen at random.
Choose $h \in H_k$ at random, and set $y = h(z)$;
Run $A'(1^k, h, y)$ and obtain output h', x';
Output x'.

Note that the distribution over the inputs of A' in the above experiment is identical to the distribution over the inputs of A' in Equation 4. For any $k \in \mathbb{N}$, $h \in H_k$ and $y \in \{0,1\}^{v(k)}$ such that $\Pr_{x \leftarrow \{0,1\}^{n(k)}}[f'_k(h,x) = (h,y)] > 0$ let:

$$\theta_h(y) \stackrel{\text{def}}{=} \frac{\min_{z \in \mathsf{image}(f_k) \bigwedge h(z)=y} \left\{\Pr_{x \leftarrow \{0,1\}^{n(k)}}[f_k(x) = z]\right\}}{\Pr_{x \leftarrow \{0,1\}^{n(k)}}[f'_k(x,h) = (h,y)]}.$$

Observe that:

$$\begin{aligned}\mathsf{Adv}_{A,\mathcal{F}}(k) &\stackrel{\text{def}}{=} \Pr_{x \leftarrow \{0,1\}^{n(k)}; z = f_k(x); x' \leftarrow A(1^k, z)}[f_k(x') = z] \\ &\geq \sum_{\hat{h},\hat{y}} \Pr_{\mathsf{Expt}_{A'}(k)}[h(f_k(x')) = y \bigwedge (h,y) = (\hat{h},\hat{y})] \cdot \theta_{\hat{h}}(\hat{y}).\end{aligned}$$

We will make use of the following standard fact (proof in full version).

Claim. Let D be a distribution over some finite domain X such that $H_2(D) \geq k$ and let ε be any positive constant, then there exists a set $B \subseteq X$ such that the following hold: (1) $\Pr_D[B] \leq 4\varepsilon$, and (2) $\forall y \notin B \ \Pr_D[y] \leq \frac{2^{1-k}}{\varepsilon}$.

Let $\varepsilon \stackrel{\text{def}}{=} \mathsf{Adv}_{A',\mathcal{F}'}(k)$. Using the previous claims we have that there exists a set $\mathsf{Bad} \subseteq (H_k \times \{0,1\}^{v(k)})$ such that:

1. $\Pr_{(h,x) \leftarrow (H_k \times \{0,1\}^n)}[f'_k(h,x) \in \mathsf{Bad}] \leq \frac{\varepsilon}{2}$
2. $\forall (h',y') \notin \mathsf{Bad} \ \Pr_{(h,x) \leftarrow (H_k \times \{0,1\}^n)}[f'_k(h,x) = (h',y')] \leq \frac{32}{\varepsilon 2^{s(k)+v(k)}}$.

Moreover, by our choice of the probability of Bad the following holds,

$$\Pr_{\mathsf{Expt}_{A'}(k)}[f'_k(h',x') = (h,y) \bigwedge (h,y) \notin \mathsf{Bad}] \geq \frac{\varepsilon}{2}.$$

Finally, by the definition of $v(k)$ the following holds for any $(h, y) \notin \mathsf{Bad}$

$$\theta_h(y) \geq \frac{\varepsilon 2^{v(k)}}{32 \cdot 2^{t_{max}(k)}} \geq \frac{\varepsilon}{32 \cdot (ck) \cdot k^m} = \frac{\varepsilon}{32 \cdot c \cdot k^{m+1}}$$

Hence:

$$\begin{aligned}
\mathsf{Adv}_{A,\mathcal{F}}(k) &\geq \sum_{(\hat{h},\hat{y}) \notin \mathsf{Bad}} \Pr_{\mathsf{Expt}_{A'}(k)}[h(f_k(x')) = y \bigwedge (h, y) = (\hat{h}, \hat{y})] \cdot \theta_{\hat{h}}(\hat{y}) \\
&\geq \frac{\varepsilon}{32 \cdot c \cdot k^{m+1}} \sum_{(\hat{h},\hat{y}) \notin \mathsf{Bad}} \Pr_{\mathsf{Expt}_{A'}(k)}[h(f_k(x')) = y \bigwedge (h, y) = (\hat{h}, \hat{y})] \\
&= \frac{\varepsilon}{32 \cdot c \cdot k^{m+1}} \Pr_{\mathsf{Expt}_{A'}(k)}[f'_k(h', x') = (h, y) \bigwedge (h, y) \notin \mathsf{Bad}] \\
&\geq \frac{\varepsilon}{32 \cdot c \cdot k^{m+1}} \cdot \frac{\varepsilon}{2} = \frac{\varepsilon^2}{64 \cdot c \cdot k^{m+1}}.
\end{aligned}$$

Since $\mathsf{Adv}_{A,\mathcal{F}}(k)$ is negligible by assumption, it must be the case that $\mathsf{Adv}_{A',\mathcal{F}'}(k)$ is negligible as well and thus $\mathcal{F}'$ is one way.

To finish the proof we show that $\mathcal{F}'$ is one-way over its range.

Claim. $\mathcal{F}'$ is one-way over its range.

Proof. Consider any PPT algorithm A'' inverting $\mathcal{F}'$ "over its range". The advantage of A'' (in this sense) is given by:

$$\begin{aligned}
\mathsf{Adv}^*_{A'',\mathcal{F}'} &\stackrel{\text{def}}{=} \Pr_{h \leftarrow H_k; y \leftarrow \{0,1\}^{v(k)}; (h', x') \leftarrow A''(1^k, h, y)}[f'_k(h', x') = (h, y)] \\
&= \frac{1}{2^{s(k)+v(k)}} \cdot \sum_{h \in H_k} \sum_{y \in \{0,1\}^{t(k)}} \Pr[A'' \text{ inverts } (h, y)],
\end{aligned}$$

where "A'' inverts (h, y)" has the obvious meaning.

Consider now the advantage of A'' in inverting $\mathcal{F}'$ in the standard sense:

$$\begin{aligned}
\mathsf{Adv}_{A'',\mathcal{F}'} &\stackrel{\text{def}}{=} \Pr_{h \leftarrow H_k; x \leftarrow \{0,1\}^{n(k)}}[A'' \text{ inverts } (h, h(f_k(x)))] \\
&= \frac{1}{2^{s(k)+n(k)}} \sum_{h \in H_k} \sum_{x \in \{0,1\}^{n(k)}} \Pr[A'' \text{ inverts } (h, h(f_k(x)))] \\
&= \frac{1}{2^{s(k)+n(k)}} \sum_{h \in H_k} \sum_{z \in \mathsf{image}(f_k)} \Pr_{x \leftarrow \{0,1\}^{n(k)}}[f_k(x) = z] \cdot \Pr[A'' \text{ inverts } (h, h(z))] \\
&\geq \frac{1}{2^{s(k)+t_{max}(k)}} \sum_{h \in H_k} \sum_{y \in \mathsf{image}(h(f_k))} \sum_{z \in h^{-1}(y)} \Pr[A'' \text{ inverts } (h, h(z))] \\
&\geq \frac{1}{2^{s(k)+t_{max}(k)}} \sum_{h \in H_k} \sum_{y \in \{0,1\}^{v(k)}} \Pr[A'' \text{ inverts } (h, y)] \\
&= \frac{2^{s(k)+v(k)}}{2^{s(k)+t_{max}(k)}} \mathsf{Adv}^*_{A'',f'} \quad \geq \quad \frac{\mathsf{Adv}^*_{A'',\mathcal{F}'}}{c \cdot k^{m+1}}.
\end{aligned}$$

Since $\mathsf{Adv}_{A'',\mathcal{F}'}$ is negligible (by the one-wayness of $\mathcal{F}'$), $\mathsf{Adv}^*_{A'',\mathcal{F}'}$ is negligible as well. This completes the proof that $\mathcal{F}'$ is one-way over its range.

6 Starting from Approximable-Preimage-Size One-Way Functions

Given an approximable-preimage-size one-way function family we first use a result by Håstad et al. [20] to transform it into a one-way function family that is "closer" to regular. From there we use the same construction of the previous section with a more careful analysis. The main result of this section is the following:

Theorem 5. *If there exists an approximable-preimage-size one-way function family then for any $0 < \delta < 1$ there exists a (δ, δ)-balanced function family which is one-way over its range.*

6.1 From Approximable to Dense

The following construction appeared in [20]:

Construction 10. *Let $\mathcal{F}=\{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}_{k\in\mathbb{N}}$ be an approximable-preimage-size one-way function family and let $\mathcal{H} = \{H_k\}$ be a 2-universal collection of hash families where each H_k is a family of functions mapping strings of length $n(k)$ to strings of length $n(k)$, and furthermore $|H_k| = 2^{s(k)}$ where $s(k) = \mathsf{poly}(k)$. Define:*

$$\hat{\mathcal{F}} = \left\{ \hat{f}_k : H_k \times \{0,1\}^{n(k)} \to H_k \times \{0,1\}^{l(k)+n(k)} \right\}_{k\in\mathbb{N}}$$

such that $\hat{f}_k(h,x) = (f_k(x), h(x)_{1\ldots(\tilde{D}_{\mathcal{F}}(f_k(x),k)+2)}, 0^{n-(\tilde{D}_{\mathcal{F}}(f_k(x),k)+2)}, h)$, where $h(x)_{1\ldots m}$ stands for the first m bits of $h(x)$.

The following lemma, proven in [20–Lemma 5.2], shows that $\hat{\mathcal{F}}$ is a family of $(s(k)+n(k)-1, s(k)+n(k))$-entropy one-way functions:

Lemma 4. *$\hat{\mathcal{F}}$ as defined in Construction 10 is one-way, and for all $k \in \mathbb{N}$, $H_2(\hat{f}_k(U_{s(k)}, U_{n(k)})) > s(k)+n(k)-1$.*

6.2 Starting from a Dense One-Way Function

Given an approximable-preimage-size one-way function family, we can transform it using Lemma 4 into a one-way function family $\mathcal{F}$ that has $(n(k)-1, n(k))$-entropy. Intuitively, such a function is "close" to being 1-regular. The following lemma shows how to use this property to construct a balanced function family which is one-way over its range.

Lemma 5. *Let $\mathcal{F} = \{f_k : \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}_{k\in\mathbb{N}}$ be an $(n(k)-1, n(k))$-entropy family of one-way functions, let $c > 24\ln 2$ be an arbitrary constant, let $\delta = (24\ln 2/c)^{1/2}$ and let $\mathcal{F}'$ be the result of applying Construction 8 with $\mathcal{F}$, $t(k) = n(k) - 1 - \log(n(k))$ and c. Then $\mathcal{F}'$ is $(2^{-k} + 12/\delta n(k), \delta)$-balanced as well as one-way over its range.*

Theorem 5 follows immediately.

Proof. (of Lemma 5) Note that since $n(k)$ is polynomial in k, there exists a constant $m \geq 0$ such that $t(k) \geq n(k) - m\log(k)$. Hence by applying Lemma 3 we have that $\mathcal{F}'$ is one-way on range. It is left to prove that $\mathcal{F}'$ is $(2^{-k}+12/\delta n(k), \delta)$-balanced. We use the following standard fact. The proof appears in the full version.

Claim. Let D be a distribution over some finite domain X such that $H_2(D) \geq k$ then for every $\varepsilon > 0$ there exists a distribution D' over X such that $H_\infty(D') \geq k - log(\frac{1}{\varepsilon})$ and $SD(D, D') \leq \varepsilon$.

Since the Renyi-entropy of $f_k(U_{n(k)})$ is at least $(n(k) - 1)$, we have that $f_k(U_{n(k)})$ is $1/n(k)$-close to having min-entropy $(n(k) - log(n(k)) - 1)$. We now apply Lemma 2 and deduce that the output distribution of f'_k, that is $(h, h(f_k(U_{n(k)})))$, is $1/n(k)$-close to a distribution that is $(2^{-k}, \delta/2)$-balanced. The proof concludes by the following claim.

Claim. Let P' be a distribution over $\{0,1\}^\ell$ that is ε-close to some distribution P that is (α, δ)-balanced. Then, P' is $((\alpha + 6\varepsilon/\delta), 2\delta)$-balanced.

Proof. Let Bad be the set of bad elements for P. Let A be the set of elements $y \notin \mathsf{Bad}$ such that $|\Pr_{P'}[y] - 1/2^\ell| > 2\delta/2^\ell$. Note that the set of bad elements Bad' of P' is a subset of $(\mathsf{Bad} \cup A)$ and therefore it is enough to bound the size and probability of this set. Note that since $A \bigcap \mathsf{Bad} = \emptyset$ we have that $\forall y \in A$ $|\Pr_P[y] - 1/2^\ell| \leq \delta/2^\ell$ and thus $|\Pr_{P'}[y] - \Pr_P[y]| > \delta/2^\ell$. Thus $SD(P', P) \geq \frac{1}{2}|A| \cdot \delta/2^\ell$. As the two distributions are ε-close, it follows that $\frac{1}{2}|A| \cdot \delta/2^\ell \leq \varepsilon$ or equivalently that $|A| \leq \frac{2\varepsilon \cdot 2^\ell}{\delta}$. Therefore we have that

$$\Pr_P[\mathsf{Bad}'] \leq \Pr_P[\mathsf{Bad} \bigcup A] \leq \Pr_P[\mathsf{Bad}] + \Pr_P[A].$$

Since for all $y \in A$ we have $\Pr_P[y] \leq (1+\delta)/2^\ell$, it follows that

$$\Pr_P[\mathsf{Bad}'] \leq \alpha + |A|(1+\delta)/2^\ell \leq \alpha + (1+\delta)\frac{2\varepsilon}{\delta} = \alpha + 2\varepsilon + \frac{2\varepsilon}{\delta}.$$

Hence:

$$\Pr_{P'}[\mathsf{Bad}'] \leq \alpha + 2\varepsilon + \frac{2\varepsilon}{\delta} + 2\varepsilon = \alpha + 4\varepsilon + \frac{2\varepsilon}{\delta} \leq \alpha + \frac{6\varepsilon}{\delta}.$$

To complete the proof we have to show that $|\mathsf{Bad}'| \leq (\alpha + \frac{6\varepsilon}{\delta})2^\ell$. But $|\mathsf{Bad}'| \leq |\mathsf{Bad}| + |A| \leq \alpha 2^\ell + \frac{2\varepsilon \cdot 2^\ell}{\delta} = (\alpha + \frac{2\varepsilon}{\delta})2^\ell$.

Acknowledgments

We are grateful to Virgil Gligor, Oded Goldreich, Danny Harnik, Omer Reingold, and Alon Rosen for helpful conversations. The third author thanks Yan Zong Ding for reading a preliminary version of this manuscript and for his encouragement. We thank the anonymous referees for comments that improved the presentation.

References

1. M. Bellare and S. Micali. How to sign given any trapdoor permutation. *J. ACM*, 39(1):214–233, 1992.
2. M. Blum. Coin flipping by phone. In *IEEE COMPCOM*, 1982.
3. M. Blum and S. Micali. How to generate cryptographically-strong sequences of pseudorandom bits. *SIAM J. Computing*, 13(4):850–864, 1984.
4. M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge. *SIAM J. Computing*, 20(6):1084–1118, 1991.
5. J.F. Boyar, S.A. Kurtz, and M.W. Krentel. Discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology*, 2(2):63–76, 1990.
6. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Computer and System Sciences*, 37(2):156–189, 1988.
7. J.L. Carter and M.N. Wegman. Universal classes of hash functions. *J. Computer and System Sciences*, 18(2):143–154, 1979.
8. I. Damgård, T. Pedersen, and B. Pfitzmann. On the existence of statistically-hiding bit commitment and fail-stop signatures. In *Crypto*, 1993.
9. U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero-knowledge proofs under general assumptions. *SIAM J. Computing*, 29(1):1–28, 1999.
10. O. Goldreich. *Foundations of Cryptography, vol. 1: Basic Tools*. Cambridge University Press, 2001.
11. O. Goldreich. *Foundations of Cryptography, vol. 2: Basic Applications*. Cambridge University Press, 2004.
12. O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In *Crypto '84*.
13. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
14. O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan, and D. Zuckerman. Security preserving amplification of hardness. In *FOCS*, 1990.
15. O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.
16. O. Goldreich, H. Krawczyk, and M. Luby. On the existence of pseudorandom generators. *SIAM J. Computing*, 22(6):1163–1175, 1993.
17. O. Goldreich and L.A. Levin. Hard-core predicates for any one-way function. In *STOC*, 1989.
18. S. Goldwasser, S. Micali, and R.L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. on Computing*, 17(2):281–308, 1988.
19. S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Crypto*, 1996.
20. J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

21. R. Impagliazzo and M. Luby. One-way functions are essential for complexity-based cryptography. In *FOCS*, 1989.
22. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, 1989.
23. Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, 2003.
24. M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.
25. M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *J. Crypto.*, 11(2):87–108, 1998.
26. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic application. In *STOC*, 1989.
27. R. Ostrovsky, R. Venkatesan, and M. Yung. Secure commitment against a powerful adversary. In *STACS*, 1992.
28. R. Ostrovsky, R. Venkatesan, and M. Yung. Fair games against an all-powerful adversary. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 13, 1993.
29. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, 1990.
30. A. Russel. Necessary and sufficient conditions for collision-free hashing. *J. Cryptology*, 8(2):87–100, 1995.
31. A. De Santis and M. Yung. On the design of provably-secure cryptographic hash functions. In *Eurocrypt*, 1990.
32. J.P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.

Smooth Projective Hashing and Two-Message Oblivious Transfer

Yael Tauman Kalai

Massachusetts Institute of Technology*
tauman@mit.edu
http://www.mit.edu/∼tauman

Abstract. We present a general framework for constructing two-message oblivious transfer protocols using a modification of Cramer and Shoup's notion of smooth projective hashing (2002). Our framework is actually an abstraction of the two-message oblivious transfer protocols of Naor and Pinkas (2001) and Aiello et al. (2001), whose security is based on the Decisional Diffie Hellman Assumption. In particular, we give two new oblivious transfer protocols. The security of one is based on the N'th-Residuosity Assumption, and the security of the other is based on both the Quadratic Residuosity Assumption and the Extended Riemann Hypothesis. Our security guarantees are not simulation based, and are similar to those of previous constructions.

When using smooth projective hashing in this context, we must deal with maliciously chosen smooth projective hash families. This raises new technical difficulties, and in particular it is here that the Extended Riemann Hypothesis comes into play.

1 Introduction

In [CS98], Cramer and Shoup introduced the first CCA2 secure encryption scheme, whose security is based on the Decisional Diffie Hellman (DDH) Assumption. They later presented an abstraction of this scheme based on a new notion which they called "smooth projective hashing" [CS02]. This abstraction yielded new CCA2 secure encryption schemes whose security is based on the Quadratic Residuosity Assumption or on the N'th Residuosity Assumption [Pa99].[1] This notion of smooth projective hashing was then used by Genarro and Lindell [GL03] in the context of key generation from humanly memorizable passwords. Analogously, their work generalizes an earlier protocol for this problem [KOY01], whose security is also based on the DDH Assumption.

In this paper, we use smooth projective hashing to construct efficient two-message oblivious transfer protocols. Our work follows the above pattern, in that

* Supported in part by NSF CyberTrust grant CNS-0430450.

[1] The N'th Residuosity Assumption is also referred to in the literature as the Decisional Composite Residuosity Assumption and as Paillier's Assumption.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 78–95, 2005.

it generalizes earlier protocols for this problem [NP01, AIR01] whose security is based on the DDH assumption. Interestingly, using smooth projective hashing in this context raises a new issue. Specifically, we must deal with maliciously chosen smooth projective hash families. This issue did not arise in the previous two applications because these were either in the public key model or in the common reference string model.

1.1 Oblivious Transfer

Oblivious transfer is a protocol between a *sender*, holding two strings γ_0 and γ_1, and a *receiver* holding a choice bit b. At the end of the protocol the receiver should learn the string of his choice (i.e., γ_b) but learn nothing about the other string. The sender, on the other hand, should learn nothing about the receiver's choice b.

Oblivious transfer, first introduced by Rabin [Rab81], is a central primitive in modern cryptography. It serves as the basis of a wide range of cryptographic tasks. Most notably, any secure multi-party computation can be based on a secure oblivious transfer protocol [Y86, GMW87, Kil88]. Oblivious transfer has been studied in several variants, all of which have been shown to be equivalent. The variant considered in this paper is the one by Even, Goldreich and Lempel [EGL85] (a.k.a. 1-out-of-2 oblivious transfer), shown to be equivalent to Rabin's original definition by Crépeau [Cre87].

The study of oblivious transfer has been motivated by both theoretical and practical considerations. On the theoretical side, much work has been devoted to the understanding of the hardness assumptions required to guarantee oblivious transfer. In this context, it is important to note that known constructions for oblivious transfer are based on relatively strong computational assumptions – either specific assumptions such as factoring or Diffie Hellman (cf. [Rab81, BM89, NP01, AIR01]) or generic assumption such as the existence of enhanced trapdoor permutations (cf. [EGL85, Gol04, Hai04]). Unfortunately, oblivious transfer cannot be reduced in a black box manner to presumably weaker primitives such as one-way functions [IR89]. On the practical side, research has been motivated by the fact oblivious transfer is considered to be the main bottleneck with respect to the amount of computation required by secure multiparty protocols. This makes the construction of efficient protocols for oblivious transfer a well-motivated task.

In particular, constructing round-efficient oblivious transfer protocols is an important task. Indeed, [NP01] (in Protocol 4.1) and [AIR01] independently constructed a *two-message* (1-round) oblivious transfer protocol based on the DDH Assumption (with weaker security guarantees than the simulation based security). Their work was the starting point of our work.

1.2 Smooth Projective Hashing

Smooth projective hashing is a beautiful notion introduced by Cramer and Shoup [CS02]. To define this notion they rely on the existence of a set X (actually a

distribution on sets), and an underlying $\mathcal{NP}$-language $L \subseteq X$ (with an associated $\mathcal{NP}$-relation R). The basic hardness assumption is that it is infeasible to distinguish between a random element in L and a random element in $X \setminus L$. This is called a *hard subset membership problem.*

A *smooth projective hash family* is a family of hash functions that operate on the set X. Each function in the family has two keys associated with it: a hash key k, and a projection key $\alpha(k)$. The first requirement (which is the standard requirement of a hash family) is that given a hash key k and an element x in the domain X, one can compute $H_k(x)$. There are two additional requirements: the "projection requirement" and the "smoothness requirement."

The "projection requirement" is that given a projection key $\alpha(k)$ and an element in $x \in L$, the value of $H_k(x)$ is uniquely determined. Moreover, computing $H_k(x)$ can be done efficiently, given the projection key $\alpha(k)$ and a pair $(x, w) \in R$. The "smoothness requirement," on the other hand, is that given a random projection key $s = \alpha(k)$ and any element in $x \in X \setminus L$, the value $H_k(x)$ is statistically indistinguishable from random.

1.3 Our Results

We present a methodology for constructing a two-message oblivious transfer protocol from any (modification of a) smooth projective hash family. In particular, we show how the previously known (DDH based) protocols of [NP01, AIR01] can be viewed as a special case of this methodology. Moreover, we show that this methodology gives rise to two new oblivious transfer protocols; one based on the N'th Residuosity Assumption, and the other based on the Quadratic Residuosity Assumption along with the Extended Riemann Hypothesis.

Our protocols, similarly to the protocols of [NP01, AIR01], are not known to be secure according to the traditional simulation based definition. Yet, they have the advantage of providing a certain level of security even against malicious adversaries without having to compromise on efficiency (see Section 3 for further discussion on the guaranteed level of security).

The basic idea. Given a smooth projective hash family for a hard subset membership problem (which generates pairs X, L according to some distribution), consider the following two-message protocol for *semi-honest* oblivious transfer. Recall that the sender's input is a pair of strings γ_0, γ_1 and the receiver's input is a choice bit b.

- $R \rightarrow S$: Choose a pair X, L (with an associated NP-relation R_L) according to the specified distribution. Randomly generate a triplet (x_0, x_1, w_b) where $x_b \in_R L$, $(x_b, w_b) \in R_L$, and $x_{1-b} \in_R X \setminus L$. Send (X, x_0, x_1).
- $S \rightarrow R$: Choose independently two random keys k_0, k_1 for $\mathcal{H}$ and send $\alpha(k_0)$ and $\alpha(k_1)$ along with $y_0 = \gamma_0 \oplus H_{k_0}(x_0)$ and $y_1 = \gamma_1 \oplus H_{k_1}(x_1)$.
- R: Retrieve γ_b by computing $y_b \oplus H_{k_b}(x_b)$, using the witness w_b and the projection key $\alpha(k_b)$.

The security of the receiver is implied by the hardness of the subset membership problem on X. Specifically, guessing the value of b is equivalent to dis-

tinguishing between a random element in L and a random element in $X \setminus L$. The security of the sender is implied by the smoothness property of the hash family $\mathcal{H}$. Specifically, given a random projection key $\alpha(k)$ and any element in $x \in X \setminus L$, the value $H_k(x)$ is statistically indistinguishable from random. Thus, the message y_{1-b} gives no information about γ_{1-b} (since $x_{1-b} \in X \setminus L$). Note that the functionality of the protocol is implied by the projection property.

Technical difficulty. Notice that when considering malicious receivers, the security of the sender is no longer guaranteed. The reason is that there is no guarantee that the receiver will choose $x_{1-b} \in X \setminus L$. A malicious receiver might choose $x_0, x_1 \in L$ and learn both values. To overcome this problem, we extend the notion of a hard subset membership problem so that it is possible to verify that at least one of x_0, x_1 belongs to $X \setminus L$. This should work even if the set X is maliciously chosen by the receiver.

It turns out that implementing this extended notion in the context of the DDH assumption is straightforward [NP01, AIR01]. Loosely speaking, in this case X is generated by choosing a random prime p, and choosing two random elements g_0, g_1 in $\mathbb{Z}_p^*$ of some prime order q. The resulting set X is defined by $X \triangleq \{(g_0^{r_0}, g_1^{r_1}) : r_0, r_1 \in \mathbb{Z}_q\}$, the corresponding language L is defined by $L \triangleq \{(g_0^r, g_1^r) : r \in \mathbb{Z}_q\}$, and the witness of each element $(g_0^r, g_1^r) \in L$ is its discrete logarithm r. In order to enable the sender to verify that two elements x_0, x_1 are not both in L, we instruct the receiver to generate x_0, x_1 by choosing at random two distinct elements $r_0, r_1 \in \mathbb{Z}_q$, setting $x_b = (g_0^{r_0}, g_1^{r_0})$, $w_b = r_0$, and $x_{1-b} = (g_0^{r_0}, g_1^{r_1})$. Notice that x_b is uniformly distributed in L, x_{1-b} is uniformly distributed in $X \setminus L$, and the sender can easily check that it is not the case that both x_0 and x_1 are in L by merely checking that they agree on their first coordinate and differ on their second coordinate.

Implementing this verifiability property in the context of the N'th Residuosity Assumption and the Quadratic Residuosity Assumption is not as easy. This part contains the bulk of technical difficulties of this work. In particular, this is where the Extended Riemann Hypothesis comes into play in the context of Quadratic Residuosity.

2 Smooth Projective Hash Functions

Our definition of smooth projective hashing differs from its original definition in [CS02]. The main difference (from both [CS02] and [GL03]) is in the definition of the smoothness requirement, which we relax to *Y-smoothness*, and in the definition of a subset membership problem, where we incorporate an additional requirement called *Y-verifiability*.

Notation. The security parameter is denoted by n. For a distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes the action of choosing x according to $\mathcal{D}$, and $x \in \mathit{support}(D)$ means that the distribution $\mathcal{D}$ samples the value x with positive probability. We denote by $x \in_R S$ the action of uniformly choosing an element from the set S. For any two

random variables X, Y, we say that X and Y are ϵ-close if $Dist(X,Y) \leq \epsilon$, where $Dist(X,Y)$ denotes the statistical difference between X and Y.[2] We say that the ensembles $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are statistically indistinguishable if there exists a negligible function $\epsilon(\cdot)$ such that for every $n \in \mathbb{N}$, the random variables X_n and Y_n are $\epsilon(n)$-close.[3] Recall that a function $\nu : \mathbb{N} \to \mathbb{N}$ is said to be negligible if for every polynomial $p(\cdot)$ and for every large enough n, $\nu(n) < 1/p(n)$.

Hard subset membership problems. A subset membership problem $\mathbf{M}$ specifies a collection $\{I_n\}_{n\in\mathbb{N}}$ of distributions, where for every n, I_n is a probability distribution over *instance descriptions*. Each instance description Λ specifies two finite non-empty sets $X, W \subseteq \{0,1\}^{poly(n)}$, and an NP-relation $R \subset X \times W$, such that the corresponding language $L \triangleq \{x : \exists w \ s.t. \ (x,w) \in R\}$ is non-empty. For every $x \in X$ and $w \in W$, if $(x,w) \in R$, we say that w is a *witness* for x. We use the following notation throughout the paper: for any instance description Λ we let $X(\Lambda)$, $W(\Lambda)$, $R(\Lambda)$ and $L(\Lambda)$ denote the sets specified by Λ.

Loosely speaking, subset membership problem $\mathbf{M} = \{I_n\}_{n\in\mathbb{N}}$ is said to be *hard* if for a random instance description $\Lambda \leftarrow I_n$, it is hard to distinguish random members of $L(\Lambda)$ from random non-members.

Definition 1 (Hard subset membership problem). *Let $\boldsymbol{M} = \{I_n\}_{n\in\mathbb{N}}$ be a subset membership problem as above. We say that $\boldsymbol{M}$ is* hard *if the ensembles $\{\Lambda_n, x_n^0\}_{n\in\mathbb{N}}$ and $\{\Lambda_n, x_n^1\}_{n\in\mathbb{N}}$ are computationally indistinguishable, where $\Lambda_n \leftarrow I_n$, $x_n^0 \in_R L(\Lambda_n)$, and $x_n^1 \in_R X(\Lambda_n) \setminus L(\Lambda_n)$.*[4]

Projective hash family. We next present the notion of a projective hash family with respect to a hard subset membership problem $\mathbf{M} = \{I_n\}_{n\in\mathbb{N}}$. Let $\mathcal{H} = \{H_k\}_{k\in K}$ be a collection of hash functions. K, referred to as the key space, consists of a set of keys such that for each instance description $\Lambda \in \mathbf{M}$,[5] there is a subset of keys $K(\Lambda) \subseteq K$ corresponding to Λ. For every Λ and for every $k \in K(\Lambda)$, H_k is a hash function from $X(\Lambda)$ to $G(\Lambda)$, where $G(\Lambda)$ is some finite non-empty set. We denote by $G = \bigcup_{\Lambda\in\mathbf{M}} G(\Lambda)$. We define a *projection key* function $\alpha : K \to S$, where S is the space of projection keys. Informally, a family $(\mathcal{H}, K, S, \alpha, G)$ is a projective hash family for $\mathbf{M}$ if for every instance description $\Lambda \in \mathbf{M}$ and for every $x \in L(\Lambda)$, the projection key $s = \alpha(k)$ uniquely determines $H_k(x)$. (We stress that the projection key $s = \alpha(k)$ is only guaranteed to determine $H_k(x)$ for $x \in L(\Lambda)$, and nothing is guaranteed for $x \in X(\Lambda) \setminus L(\Lambda)$.)

---

[2] Recall that $Dist(X,Y) \triangleq \frac{1}{2}\sum_{s\in S} |Pr[X = s] - Pr[Y = s]|$, or equivalently, $Dist(X,Y) \triangleq \max_{S'\subset S} |Pr[X \in S'] - Pr[Y \in S']|$, where S is any set that contains the support of both X and Y.

[3] For simplicity, throughout this paper we say that two random variables X_n and Y_n are statistically indistinguishable, meaning that the corresponding distribution ensembles $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are statistically indistinguishable.

[4] Note that this hardness requirement also implies that it is hard to distinguish between a random element $x \in_R L(\Lambda)$ and a random element $x \in_R X(\Lambda)$. We will use this fact in the proof of Theorem 1.

[5] We abuse notation and let $\Lambda \in \mathbf{M}$ denote the fact that $\Lambda \in support(I_n)$ for some n.

Definition 2 (Projective hash family). *$(\mathcal{H}, K, S, \alpha, G)$ is a projective hash family for a subset membership problem $\boldsymbol{M}$ if for every instance description $\Lambda \in \boldsymbol{M}$ there is a well defined (not necessarily efficient) function f such that for every $x \in L(\Lambda)$ and every $k \in K(\Lambda)$, $f(x, \alpha(k)) = H_k(x)$.*

Efficient projective hash family. We say that a projective hash family is efficient if there exist polynomial time algorithms for: (1) Sampling a key $k \in_R K(\Lambda)$ given Λ; (2) Computing a projection $\alpha(k)$ from Λ and $k \in K(\Lambda)$; (3) Computing $H_k(x)$ from Λ, $k \in K(\Lambda)$ and $x \in X(\Lambda)$; and (4) Computing $H_k(x)$ from Λ, $(x, w) \in R(\Lambda)$ and $\alpha(k)$, where $k \in K(\Lambda)$. Notice that this gives two ways to compute $H_k(x)$: either by knowing the hash key k, or by knowing the projection key $\alpha(k)$ and a witness w for x.

Y-smooth projective hash family. Let Y be any function from instance descriptions $\Lambda \in \mathbf{M}$ to subsets $Y(\Lambda) \subseteq X(\Lambda) \setminus L(\Lambda)$. Loosely speaking, a projective hash family for $\mathbf{M}$ is Y-smooth if for every instance description $\Lambda = (X, W, R)$, for every $x \in Y(\Lambda)$, and for a random $k \in_R K(\Lambda)$, the projection key $\alpha(k)$ reveals (almost) nothing about $H_k(x)$.

Definition 3 (Y-smooth projective hash family). *A projective hash family $(\mathcal{H}, K, S, \alpha, G)$ for a subset membership problem $\boldsymbol{M}$ is said to be* Y-smooth *if for every (even maliciously chosen) instance description $\Lambda = (X, W, R)$ and every $x \in Y(\Lambda)$, the random variables $(\alpha(k), H_k(x))$ and $(\alpha(k), g)$ are statistically indistinguishable, where $k \in_R K(\Lambda)$ and $g \in_R G(\Lambda)$.*[6]

A Y-smooth projective hash family thus has the property that a projection of a (random) key enables the computation of $H_k(x)$ for $x \in L$, but gives almost no information about the value of $H_k(x)$ for $x \in Y(\Lambda)$.

Remark. This definition of Y-smooth projective hash family differs from the original definition proposed in [CS02] in two ways. First, it requires the smoothness property to hold against *maliciously* chosen instance descriptions Λ, whereas in [CS02] the smoothness is only with respect to $\Lambda \in \mathbf{M}$. Second, it requires the smoothness property to hold with respect to every $x \in Y$, whereas in [CS02] the smoothness condition is required to hold for randomly chosen $x \in_R X \setminus L$.

The main reason for our divergence from the original definition in [CS02] is that we need to cope with maliciously chosen Λ. We would like to set $Y = X \setminus L$ (as in [CS02]), and construct a $(X \setminus L)$-smooth projective hash family. However, we do not know how to construct such a family, for which the smoothness condition holds for *every* (even maliciously chosen) Λ.[7] Therefore, we relax our smoothness requirement and require only Y-smoothness, for some

[6] We assume throughout this paper, without loss of generality, that a (maliciously chosen) Λ has the same structure as an honestly chosen Λ.

[7] We note that [CS02, GL03] did not deal with maliciously chosen Λ's, and indeed the smoothness property of their constructions does not hold for maliciously chosen Λ's.

$Y \subseteq X \setminus L$. In both our constructions of Y-smooth projective hash families, $Y(\Lambda) \subset X(\Lambda) \setminus L(\Lambda)$ for maliciously chosen $\Lambda \notin \mathbf{M}$, and $Y(\Lambda) = X(\Lambda) \setminus L(\Lambda)$ for every honestly chosen $\Lambda \in \mathbf{M}$. Jumping ahead, the latter will enable the (honest) receiver to choose $x_b \in_R L(\Lambda)$, $x_{1-b} \in_R X(\Lambda) \setminus L(\Lambda)$ such that x_{1-b} is also in $Y(\Lambda)$. This will enable the (honest) sender to be convinced of its security by checking that either x_0 or x_1 is in $Y(\Lambda)$, and it will enable the (honest) receiver to be convinced that a (dishonest) sender cannot guess the bit b, assuming the underlying subset membership problem is hard. (From now on the reader should think of $Y(\Lambda)$ as equal to $X(\Lambda) \setminus L(\Lambda)$ for every $\Lambda \in \mathbf{M}$.)

Thus, we need a subset membership problem $\mathbf{M}$ such that for every honestly chosen $\Lambda \in \mathbf{M}$ it is easy to sample uniformly from both $L(\Lambda)$ and $X(\Lambda) \setminus L(\Lambda)$. On the other hand, for every (even maliciously chosen) (Λ, x_0, x_1) it is easy to verify that either $x_0 \in Y(\Lambda)$ or $x_1 \in Y(\Lambda)$. To this end we define the notion of a "Y-verifiably samplable" subset membership problem.

Definition 4 (Y-verifiably samplable subset membership problem). *A subset membership problem $\boldsymbol{M} = \{I_n\}_{n\in\mathbb{N}}$ is said to be* Y-verifiably samplable *if the following conditions hold.*

1. Problem samplability: *There exists a probabilistic polynomial-time algorithm that on input 1^n, samples an instance $\Lambda = (X, W, R)$ according to I_n.*
2. Member samplability: *There exists a probabilistic polynomial-time algorithm that on input an instance description $\Lambda = (X, W, R) \in \boldsymbol{M}$, outputs an element $x \in L$ together with its witness $w \in W$, such that the distribution of x is statistically close to uniform on L.*
3. Non-member samplability: *There exists a probabilistic polynomial-time algorithm $\mathcal{A}$ that given an instance description $\Lambda = (X, W, R) \in \boldsymbol{M}$ and an element $x_0 \in X$, outputs an element $x_1 = \mathcal{A}(\Lambda, x_0)$, such that if $x_0 \in_R L$ then the distribution of x_1 is statistically close to uniform on $X \setminus L$, and if $x_0 \in_R X$ then the distribution of x_1 is statistically close to uniform on X.*
4. Y-Verifiability: *There exists a probabilistic polynomial-time algorithm $\mathcal{B}$, that given* any *triplet (Λ, x_0, x_1), verifies that there exists a bit b such that $x_b \in Y(\Lambda)$. This should hold even if Λ is maliciously chosen. Specifically:*
   - *For every Λ and every x_0, x_1, if both $x_0 \notin Y(\Lambda)$ and $x_1 \notin Y(\Lambda)$ then $\mathcal{B}(\Lambda, x_0, x_1) = 0$.*
   - *For every honestly chosen $\Lambda \in \boldsymbol{M}$ and every x_0, x_1, if there exists b such that $x_b \in L(\Lambda)$ and $x_{1-b} \in support(\mathcal{A}(\Lambda, x_b))$, then $\mathcal{B}(\Lambda, x_0, x_1) = 1$.*

For simplicity, throughout the paper we do not distinguish between uniform and statistically close to uniform distributions. This is inconsequential.

3 Security of Oblivious Transfer

Our definition of oblivious transfer is similar to the ones considered in previous works on oblivious transfer in the Bounded Storage Model [DHRS04, CCM98].

A similar (somewhat weaker) definition was also used in [NP01] in the context of their DDH based two message oblivious transfer protocol.

In what follows we let $view_{\hat{S}}(\hat{S}(z), R(b))$ denote the view of a cheating sender $\hat{S}(z)$ after interacting with $R(b)$. This view consists of its input z, its random coin tosses, and the messages that it received from $R(b)$ during the interaction. Similarly, we let $view_{\hat{R}}(S(\gamma_0, \gamma_1), \hat{R}(z))$ denote the view of a cheating Receiver $\hat{R}(z)$ after interacting with $S(\gamma_0, \gamma_1)$.

Definition 5 (Secure implementation of Oblivious Transfer). *A two party protocol (S, R) is said to* securely implement oblivious transfer *if it is a protocol in which both the sender and the receiver are probabilistic polynomial time machines that get as input a security parameter n in unary representation. Moreover, the sender gets as input two strings $\gamma_0, \gamma_1 \in \{0,1\}^{\ell(n)}$, the receiver gets as input a choice bit $b \in \{0,1\}$, and the following conditions are satisfied:*

- Functionality: *If the sender and the receiver follow the protocol then for any security parameter n, any two input strings $\gamma_0, \gamma_1 \in \{0,1\}^{\ell(n)}$, and any bit b, the receiver outputs γ_b whereas the sender outputs nothing.*[8]
- Receiver's security: *For any probabilistic polynomial-time adversary $\hat{S}$, executing the sender's part, for any security parameter n, and for any auxiliary input z of size polynomial in n, the view that $\hat{S}(z)$ sees when the receiver tries to obtain the first message is computationally indistinguishable from the view it sees when the receiver tries to obtain the second message. That is,*

$$\{view_{\hat{S}}(\hat{S}(z), R(1^n, 0))\}_{n,z} \stackrel{\mathsf{c}}{\equiv} \{view_{\hat{S}}(\hat{S}(z), R(1^n, 1))\}_{n,z}$$

- Sender's security: *For any deterministic (not necessarily polynomial-time) adversary $\hat{R}$, executing the receiver's part, for any security parameter n, for any auxiliary input z of size polynomial in n, and for any $\gamma_0, \gamma_1 \in \{0,1\}^{\ell(n)}$, there exists a bit b such that for every $\psi \in \{0,1\}^{\ell(n)}$, the view of $\hat{R}(z)$ when interacting with $S(1^n, \gamma_b, \psi)$, and the view of $\hat{R}(z)$ when interacting with $S(1^n, \gamma_0, \gamma_1)$, are statistically indistinguishable.*[9] *That is,*

$$\{view_{\hat{R}}(S(1^n, \gamma_0, \gamma_1), \hat{R}(z))\}_{n,\gamma_0,\gamma_1,z} \stackrel{\mathsf{s}}{\equiv} \{view_{\hat{R}}(S(1^n, \gamma_b, \psi), \hat{R}(z))\}_{n,\gamma_b,\psi,z}$$

Note that Definition 5 (similarly to the definitions in [DHRS04, NP01]) departs from the traditional, simulation based, definition in that it handles the security of the sender and of the receiver separately. This results in a somewhat weaker security guarantee, with the main drawback being that neither the sender nor the receiver are actually guaranteed to "know" their own input. (This is unavoidable in two message protocols using "standard" techniques).

It is easy to show that Definition 5 implies simulatability for semi honest adversaries (the proof is omitted due to lack of space). More importantly, Definition 5 also gives meaningful security guarantees in face of malicious participants.

[8] This condition is also referred to as the completeness condition.

[9] We abuse notation by letting $S(1^n, \gamma_b, \psi)$ denote $S(1^n, \gamma_0, \psi)$ if $b = 0$, and letting it denote $S(1^n, \psi, \gamma_1)$ if $b = 1$.

In the case of a malicious sender, the guarantee is that the damage incurred by malicious participation is limited to "replacing" the input strings γ_0, γ_1 with a pair of strings that are somewhat "related" to the receiver's first message (without actually learning anything about the receiver's choice). In the case of a malicious receiver, Definition 5 can be shown to provide exponential time simulation of the receiver's view of the interaction (similarly to the definition of [NP01]). In particular, the interaction gives no information to an unbounded receiver beyond the value of γ_b. (Again, the proof is omitted due to lack of space.)

4 Constructing 2-Round OT Protocols

Let $\mathbf{M} = \{I_n\}_{n\in\mathbb{N}}$ be a hard subset membership problem which is Y-verifiably samplable, and let $(\mathcal{H}, K, S, \alpha, G)$ be a an efficient Y-smooth projective hash family for $\mathbf{M}$. Recall that the Y-verifiably samplable condition of $\mathbf{M}$ implies the existence of algorithms $\mathcal{A}$ and $\mathcal{B}$ as described in Section 2.

We assume for simplicity that for any n and for any $\Lambda \in I_n$, $G(\Lambda) = \{0,1\}^{\ell(n)}$, and that the two messages γ_0, γ_1, to be transferred in the OT protocol, are binary strings of length at most $\ell(n)$. Let n be the security parameter. Let (γ_0, γ_1) be the input of the sender and let $b \in \{0,1\}$ be the input of the receiver.

$R \to S$: The receiver chooses a random instance description $\Lambda = (X, W, R) \leftarrow I_n$. It then samples a random element $x_b \in_R L$ together with its corresponding witness w_b, using the member samplability algorithm, and invokes Algorithm $\mathcal{A}$ on input (Λ, x_b) to obtain a random element $x_{1-b} \in X \setminus L$. It sends (Λ, x_0, x_1).

$S \to R$: The sender invokes algorithm $\mathcal{B}$ on input (Λ, x_0, x_1) to verify that there exists a bit b such that $x_{1-b} \in Y(\Lambda)$. If $\mathcal{B}$ outputs 0 then it aborts, and if $\mathcal{B}$ outputs 1 then it chooses independently at random $k_0, k_1 \in_R K(\Lambda)$, and sends $\alpha(k_0)$ and $\alpha(k_1)$ along with $y_0 = \gamma_0 \oplus H_{k_0}(x_0)$ and $y_1 = \gamma_1 \oplus H_{k_1}(x_1)$.

R: The receiver retrieves γ_b by computing $y_b \oplus H_{k_b}(x_b)$ using the projection key $\alpha(k_b)$ and the pair (x_b, w_b).

We next prove that the above protocol is secure according to Definition 5. Intuitively, the receiver's security follows from the fact that x_b is uniformly distributed in L, x_{1-b} is uniformly distributed in $X \setminus L$, and from the assumption that it is hard to distinguish random L elements from random $X \setminus L$ elements. The sender's security follows from the assumption that $(\mathcal{H}, K, S, \alpha, G)$ is a Y-smooth projective hash family for $\mathbf{M}$, and from the assumption that one of x_0 or x_1 is in $Y(\Lambda)$ (otherwise, it will be detected by $\mathcal{B}$ and the sender will abort).

Theorem 1. *The above 2-round OT protocol is secure according Definition 5, assuming $\boldsymbol{M}$ is a Y-verifiably samplable hard subset membership problem, and assuming $(\mathcal{H}, K, S, \alpha, G)$ is a Y-smooth projective hash family for $\boldsymbol{M}$.*

Proof. we start by proving the receiver's security. Assume for the sake of contradiction that there exists a (malicious) probabilistic polynomial-time sender $\hat{S}$

such that for infinitely many n's there exists a polynomial size auxiliary input z_n such that $\hat{S}(z_n)$ can predict (with non-negligible advantage) the choice bit b when interacting with $R(1^n, b)$. In what follows, we use $\hat{S}(z_n)$ to break the hardness of $\mathbf{M}$, by distinguishing between $x \in_R L$ and $x \in_R X$. Given an instance description $\Lambda = (X, W, R) \leftarrow (I_n)$ and an element $x \in X$:

1. Choose at random a bit b and let $x_b = x$
2. Apply algorithm $\mathcal{A}$ on input (Λ, x_b) to obtain an element x_{1-b}.
3. Feed $\hat{S}(z_n)$ the message (Λ, x_0, x_1), and obtain its prediction bit b'.
4. If $b' = b$ then predict "$x \in_R L$" and if $b' \neq b$ then predict "$x \in_R L$."

Notice that if $x_b \in_R L$ then $\hat{S}(z_n)$ will predict the bit b with non-negligible advantage (follows from our contradiction assumption). On the other hand, if $x_b \in_R X$ then x_{1-b} is also uniformly distributed in X. In this case it is impossible (information theoretically) to predict b.

We now turn to prove the sender's security. Let $\hat{R}$ be any (not necessarily polynomial time) malicious receiver, and for any $n \in \mathbb{N}$, let z_n be any polynomial size auxiliary information given to $\hat{R}$. Let (Λ_n, x_0, x_1) be the first message sent by $\hat{R}(z_n)$. Our goal is to show that for every $n \in \mathbb{N}$ and for every $\gamma_0, \gamma_1 \in \{0,1\}^{\ell(n)}$, there exists $b \in \{0,1\}$ such that the random variables $view_{\hat{R}}(S(1^n, \gamma_0, \gamma_1), \hat{R}(z_n))$ and $view_{\hat{R}}(S(1^n, \gamma_b, \psi), \hat{R}(z_n))$ are statistically indistinguishable.

We assume without loss of generality that either $x_0 \in Y(\Lambda_n)$ or $x_1 \in Y(\Lambda_n)$. If this is not the case, the sender aborts the execution and b can be set to either 0 or 1. Let b be the bit satisfying $x_{1-b} \in Y(\Lambda_n)$. By the Y-smoothness property of the hash family, the random variables $(\alpha(k), H_k(x_{1-b}))$ and $(\alpha(k), g)$ are statistically indistinguishable, for a random $k \in_R K(\Lambda_n)$ and a random $g \in_R G(\Lambda_n)$. This implies that the random variables $(\alpha(k), \gamma_{1-b} \oplus H_k(x_{1-b}))$ and $(\alpha(k), g)$ are statistically indistinguishable, which implies that $view_{\hat{R}}(S(1^n, \gamma_0, \gamma_1), \hat{R}(z))$ and $view_{\hat{R}}(S(1^n, \gamma_b, \psi), \hat{R}(z))$ are statistically indistinguishable.

5 Constructing Smooth Projective Hash Families

We next present two constructions of Y-smooth projective hash families for hard subset membership problems which are Y-verifiably samplable. One based on the N'th Residuosity Assumption, and the other based on the Quadratic-Residuosity Assumption together with the Extended Reimann Hypothesis. A key vehicle in both constructions is the notion of an (ϵ, Y)-universal projective hash family.

Definition 6 (Universal projective hash families). *Let $\boldsymbol{M} = \{I_n\}_{n \in \mathbb{N}}$ be any hard subset membership problem. A projective hash family $(\mathcal{H}, K, S, \alpha, G)$ for $\boldsymbol{M}$ is said to be* (ϵ, Y)-universal *if for every n, every (maliciously chosen) Λ corresponding to the security parameter n, every $x \in Y(\Lambda)$ and every $g \in G(\Lambda)$, $Pr_{k \in_R K(\Lambda)}[H_k(x) = g \mid \alpha(k)] \leq \epsilon(n)$.*

As shown in [CS02], it is possible to reduce the error rate in a (ϵ, Y)-universal projective hash family from ϵ to ϵ^t (via independent repetitions). Once the error

rate is reduced to be a negligible function ϵ^t, it is possible to transform the (ϵ^t, Y)-universal projective hash family into a Y-smooth projective hash family by applying the Leftover Hash Lemma. Both transformations preserve efficiency (up to polynomial factors). Due to lack of space we omit the details of these transformations, and we refer the interested reader to [CS02].

We conclude that it suffices to construct subset membership problems which are Y-verifiably samplable and for which there exists an efficient $(\frac{1}{2}, Y)$-universal projective hash family. In the remaining of this paper we present two such constructions – the first based on the N'th Residuosity Assumption, and the second based on the Quadratic-Residuosity Assumption together with the Extended Reimann Hypothesis.

5.1 *N*'th Residuosity Assumption

The N'th Residuosity Assumption. Let p, q be distinct safe primes; namely $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ are odd primes. Let $N = pq$ and let J_{N^2} be the subgroup of $\mathbb{Z}^*_{N^2}$, consisting of all elements with Jacobi symbol 1. Let P be the subgroup consisting of all N'th powers of elements in J_{N^2}. The N'th Residuosity Assumption, originally introduced by Paillier [Pa99], asserts that given only N, it is hard to distinguish random elements of J_{N^2} from random elements of P.[10,11]

Overview of the constructions under the N'th Residuosity Assumption. We would like to use the constructions given in [CS02]. They construct a subset membership problem that generates instances where $X = J_{N^2}$ and $L = P$ (so that the hardness property follows from the N'th Residuosity Assumption). They define a corresponding universal projective hash family by $H_k(x) = x^k (\text{mod } N^2)$, with the projection key of k being $\alpha(k) = g^{Nk} (\text{mod } N^2)$, where $g^N (\text{mod } N^2)$ is an a priori chosen generator for L. In their proof of the universal property, they make strong use of the fact that for honestly chosen N's (N's which are a product of two safe primes), P can also be characterized by $P = \{x \in J_{N^2} : order(x) \text{ is co-prime to } N\}$. In our case we must also consider maliciously chosen N's, in which case this characterization does not remain true.

In order to ensure that for every N (even maliciously chosen), it still holds that every element in L is of order which is co-prime to N, we change the definition of L: rather than taking L to be all the N'th powers elements in J_{N^2}, we take L to be all the T'th powers elements in J_{N^2}, where $T \triangleq N^{\lceil \log N \rceil + 1}$. As we shall see shortly, this ensures that for every (even maliciously chosen) N, every element in L is of order which is co-prime to N, and for every honestly

[10] Actually, Paillier did not make the restriction to safe primes or to elements in J_{N^2}. We note that the N'th Residuosity Assumption without these restrictions implies the N'th Residuosity Assumption with these restrictions, assuming that safe primes are sufficiently dense, as we do here. We refer the reader to [CS02] for more details.

[11] Jumping ahead, the reason that we restrict our attention to elements in J_{N^2} is that this results with the subgroup L being cyclic. This is an important point that will be elaborated on below.

chosen N, this new L is equal to the previous one, and thus it remains hard to distinguish random X elements from random L elements, under the N'th Residuosity Assumption.

The subset membership problem M.

1. *Problem samplability:* For every n, I_n is a samplable distribution that generates an instance description Λ as follows: On input 1^n,
   (a) Generate two random n bit safe primes p, q; namely, primes p and q such that $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ are odd primes. Let $N = pq$, $N' = p'q'$, and $T \triangleq N^{\lceil \log N \rceil + 1}$.
   (b) Generate a random (non-square) element $g \in \mathbb{Z}^*_{N^2}$ with Jacobi symbol 1, by choosing a random element $\mu \in_R \mathbb{Z}^*_{N^2}$ and setting $g = -\mu^2 (\text{mod } N^2)$.[12]
   (c) Output $\Lambda = (N, \mu)$, which specifies (X, W, R) in the following way: $X \triangleq J_{N^2}$, $L \triangleq \langle g^T \rangle$ is the subgroup generated by $g^T (\text{mod } N^2)$, $W \triangleq \{0, 1, \ldots, \lfloor N/2 \rfloor\}$, and $R \triangleq \{(g^{Tr}, r) : r \in W\}$.

   Notice that for every (even maliciously chosen) N, it holds that $L \subseteq \{x \in J_{N^2} : order(x)$ is co-prime to $N\}$. This is the case since the order of g divides $N\phi(N)$ (which is the order of $Z^*_{N^2}$), p and q divide $\phi(N)$ at most $\lceil \log N \rceil$ times, and they divide N exactly once. Thus p and q divide $N\phi(N)$ at most $\lceil \log N \rceil + 1$ times, and thus they divide the order of g at most $\lceil \log N \rceil + 1$ times. This implies that the order of $g^T (\text{mod } N^2)$ (where $T = N^{\lceil \log N \rceil + 1}$) is co-prime to both p and q, and thus is co-prime to N.

   Moreover, for every honestly chosen N, with overwhelming probability $L = P = \{x \in J_{N^2} : order(x)$ is co-prime to $N\}$. This follows from the fact that $|J_{N^2}| = N\phi(N)/2 = 2NN'$, which implies that P is a cyclic group of order $2N'$. Thus, for any random non-square element g in J_{N^2}, g^N is a generator of P with overwhelming probability. Moreover, since the order of g^N is co-prime to N, it follows that $\langle g^T \rangle = \langle g^N \rangle$.

   Let $Y(\Lambda) = \{x \in J_{N^2} : order(x)$ is *not* co-prime to $N\}$.[13]
2. *Member samplability:* On input $\Lambda = (N, \mu)$, choose a random $r \in_R W$ and output $g^{Tr} \in L(\Lambda)$ together with its corresponding witness r, where $g = -\mu^2 (\text{mod } N^2)$
3. *Non-member samplability:* On input $\Lambda = (N, \mu)$ and $x \in L(\Lambda)$, $\mathcal{A}$ chooses a random $a \in_R \{1, \ldots, N-1\}$, and outputs $x(1 + aN) \in X(\Lambda) \setminus L(\Lambda)$.

   Notice that for every $a \in \{1, ..., N-1\}$, $order(1 + aN)$ divides N (and is different than 1), which implies that $1 + aN \in Y(\Lambda)$.
4. *Y-Verifiability:* On input $(1^n, \Lambda, x_0, x_1)$, $\mathcal{B}$ outputs 1 if and only if $x_0, x_1 \in J_{N^2}$, $x_0 \neq x_1$, and $(x_0/x_1)^N = 1 (mod\ N^2)$.

The fact that **M** is a hard subset membership problem follows from the N'th Residuosity Assumption and from the fact that for every honestly chosen $\Lambda \in \mathbf{M}$, with overwhelming probability $L(\Lambda) = P$.

[12] Recall that for N which is a product of two safe primes $-1 \in J_N \setminus QR_N$.

[13] Notice that for every (even maliciously chosen) Λ, it holds that $Y(\Lambda) \subseteq X \setminus L$, and for honestly chosen Λ it holds that $Y(\Lambda) = X(\Lambda) \setminus L(\Lambda)$ with overwhelming probability.

We next show that $\mathbf{M}$ is Y-verifiably samplable, under the Nth Residuosity Assumption. Fix any $\Lambda = (N, \mu) \in \mathbf{M}$. It is easy to see that the member samplability algorithm samples a random element in L. Moreover, notice that $X = P \cdot H \triangleq \{x \cdot y : x \in P, y \in H\}$, where $H \triangleq \langle 1 + N \rangle$. This is the case since for every N which is a product of two safe primes, it holds that $P \cap H = \{1\}$ (since the order of elements in P divide $2N'$, the order of elements in H divide N, and $GCD(2N', N) = 1$). This implies that $|P \cdot H| = |P| \cdot |H| = 2N'N$, which together with the fact that $P \cdot H \subseteq J_{N^2}$ implies that $P \cdot H = J_{N^2}$. Now, recall that $\mathcal{A}(\Lambda, x) = x(1 + aN)$ for some uniformly chosen $a \in \{1, \ldots, N-1\}$. Thus, if $x \in_R X$ then $\mathcal{A}(\Lambda, x) \in_R X$, and if $x \in_R L$ then $\mathcal{A}(\Lambda, x) \in_R X \setminus L$, which implies that the non-member samplability requirement holds.

It remains to show that the Y-verifiability requirement holds. Notice that for every (even maliciously chosen) N and for every $x \neq 1$ such that $x^N = 1 (\text{mod } N^2)$, it holds that $x \in Y(\Lambda)$. Thus, for every distinct x_0, x_1, if $(x_0/x_1)^N = 1 (\text{mod } N^2)$ then $x_0/x_1 \in Y(\Lambda)$, which implies that either $x_0 \in Y(\Lambda)$ or $x_1 \in Y(\Lambda)$.

$(\frac{1}{2}, Y)$-Universal Projective Hashing for $\mathbf{M}$. Consider the projective hash family $(\mathcal{H}, K, S, \alpha, G)$, defined as follows. For every $\Lambda = (N, \mu) \in \mathbf{M}$:

- Let $K(\Lambda) = \{0, 1, \ldots, \lfloor \frac{N^2}{2} \rfloor\}$ and let $K = \bigcup_{\Lambda \in \mathbf{M}} K(\Lambda)$.
- Let $G(\Lambda) = J_{N^2}$ and let $G = \bigcup_{\Lambda \in \mathbf{M}} G(\Lambda)$.
- For every $k \in K(\Lambda)$, let $H_k(x) = x^k (\text{mod } N^2)$.
- For every $k \in K(\Lambda)$, let $\alpha(k) = g^{Tk} (\text{mod } N^2)$, where $T \triangleq N^{\lceil \log N \rceil + 1}$ and $g = -\mu^2 (\text{mod } N^2)$.

Claim. $(\mathcal{H}, K, S, \alpha, G)$ is an efficient $(\frac{1}{2}, Y)$-universal projective hash family for $\mathbf{M}$.

Proof. It is straightforward to verify that all the efficiency requirements hold. As for the projection requirement, this follows from the fact that for every $k \in K(\Lambda)$ and every $x = g^{Tr} (mod\ N^2) \in L(\Lambda)$,

$$H_k(x) = x^k (mod\ N^2) = (g^{Tr})^k (mod\ N^2) = (g^{Tk})^r (mod\ N^2) = \alpha(k)^r (mod\ N^2).$$

We next show that it is $(\frac{1}{2}, Y)$-universal. Fix any (even maliciously chosen) $\Lambda = (N, \mu)$, and let $Z \triangleq \phi(N^2)/GCD(\phi(N^2), T)$. Notice that $GCD(N, Z) = 1$, which implies that for every $y \in Y(\Lambda)$, $y^Z \neq 1 (mod\ N^2)$ (since the order of y is *not* co-prime to N). Also notice that for every hash key k, $\alpha(k) = \alpha(k + Z)$. Fix any $y \in Y(\Lambda)$. Since for every s there are at least two elements $k, k + Z \in K(\Lambda)$ such that $s = \alpha(k) = \alpha(k + Z)$, and since $y^Z \neq 1$, it follows that s does not uniquely determine $H_k(y)$, implying that $(\mathcal{H}, K, S, \alpha, G)$ is a $(\frac{1}{2}, Y)$-universal projective hash family.

5.2 The Quadratic Residuosity Assumption

The Quadratic Residuosity Assumption. Let p, q be distinct safe primes; namely, $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ are odd primes. Let $N = pq$, let J_N be the

subgroup of $\mathbb{Z}_N^*$ consisting of all elements with Jacobi symbol 1, and let QR_N be the subgroup of $\mathbb{Z}_N^*$ consisting of all quadratic residues (note that $QR_N \subseteq J_N$). The Quadratic Residuosity Assumption asserts that given only N, it is hard to distinguish random elements of J_N from random elements of QR_N.

Overview of the constructions under the Quadratic Residuosity Assumption. We would like to use the constructions given in [CS02]. They construct a subset membership problem that generates instances where $X = J_N$ and $L = QR_N$ (so that the hardness property follows from the Quadratic Residuosity Assumption). They define a corresponding universal projective hash family by $H_k(x) = x^k (\text{mod } N)$, with the projection key of k being $\alpha(k) = g^{2k} (\text{mod } N)$, where $g^2 (\text{mod } N)$ is an a priori chosen generator for L. In their proof of the universal property, they make strong use of the fact that for honestly chosen N's (N's which are a product of two safe primes), QR_N can also be characterized by $QR_N = \{x \in J_N : order(x) \text{ is odd}\}$. In our case we must also consider maliciously chosen N's, in which case this characterization does not remain true.

In order to ensure that for every N (even maliciously chosen), it still holds that every element in L is of odd order, we change the definition of L: rather than taking L to be the set of all squares in J_N, we take L to be the set of all the T'th powers elements in J_N, where $T \triangleq 2^{\lceil \log N \rceil}$. As we shall see shortly, this ensures that for every (even maliciously chosen) N, every element in L is of odd order, and for every honestly chosen N, this new L is equal to the previous one, and thus it remains hard to distinguish random X elements from random L elements, under the Quadratic Residuosity Assumption.

We would like to prove that this subset membership problem, which generates instances with $X = J_N$ and $L = QR_N$ (with overwhelming probability for honestly chosen N's), is Y-verifiably samplable for some $Y \subseteq J_N \setminus QR_N$. However, achieving the non-member samplability property is quite problematic. The crux of the problem is that we cannot efficiently sample an element in $J_N \setminus QR_N$ for maliciously chosen N's.[14] What we do know (under the Extended Reimann Hypothesis) is how to sample $\log^3 N$ elements such that at least one of them is in $J_N \setminus QR_N$ (though we don't know which one).[15] Thus, rather than constructing a Y-verifiably samplable subset membership problem, which is associated with a single algorithm $\mathcal{A}$ for sampling a non-member element, we will construct a subset membership problem with many ($t = \log^3 N$) algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_t$, with the guarantee that *at least one* of them is actually sampling a non-member element. Correspondingly, there will be many verification algorithms $\mathcal{B}_1, \ldots, \mathcal{B}_t$, with the guarantee that for every i it holds that $\mathcal{B}_i(\Lambda, x, \mathcal{A}_i(x)) = 1$, and that

[14] Indeed, for N's that are a product of two safe primes $-1 \in J_N \setminus QR_N$, but this is not guaranteed in general.

[15] There is a subtle issue here. The above statement is not true if N is a power of a single prime (i.e., if N is of the form $N = p^\alpha$, for some prime p and some $\alpha \geq 1$), since in this case $J_N \setminus QR_N = \emptyset$. Fortunately, we can assume from now on (without loss of generality) that N is never of that form, since this can be checked in polynomial time.

at least one of the $\mathcal{B}_i$'s outputs 1 on input (Λ, x_0, x_1) only if either $x_0 \in Y(\Lambda)$ or $x_1 \in Y(\Lambda)$.

The idea would be to use this subset membership problem to construct an oblivious transfer protocol as follows:

$R \to S$: On input $b \in \{0,1\}$, the receiver chooses a random instance description Λ together with t pairs $(x_0^1, x_1^1), \ldots, (x_0^t, x_1^t)$, and corresponding t witnesses $w_b^1, \ldots, w_b^t$, such that for each $i \in \{1, \ldots, t\}$ it holds that $x_b^i \in_R L(\Lambda)$, $(x_b^i, w_b^i) \in R(\Lambda)$, and $x_{1-b}^i = \mathcal{A}_i(\Lambda, x_b^i)$. It sends $(x_0^1, x_1^1), \ldots, (x_0^t, x_1^t)$.

$S \to R$: The sender first checks that $\mathcal{B}_i(\Lambda, x_0^i, x_1^i) = 1$ for all $i \in \{1, \ldots, t\}$. If this check does not pass then he aborts. If the check does pass then the sender splits his input (γ_0, γ_1) into t random shares $(\gamma_0^1, \gamma_1^1) \ldots, (\gamma_0^t, \gamma_1^t)$. He then chooses t pairs of random hash keys $(k_0^1, k_1^1), \ldots (k_0^t, k_1^t)$, and sends for each $i \in \{1, \ldots, t\}$ the projection keys $\alpha(k_0^i)$ and $\alpha(k_1^i)$ together with the values $y_0^i = H_{k_0^i}(x_0^i) \oplus \gamma_0^i$ and $y_1^i = H_{k_1^i}(x_1^i) \oplus \gamma_1^i$.

R: The receiver retrieves γ_b by computing $y_b^i \oplus H_{k_b^i}(x_b^i)$, using the projection key $\alpha(k_b^i)$ and the pair (x_b^i, w_b^i), and by computing the XOR of all these values.

The sender's security is ensured since we know (under the Extended Reimann Hypothesis) that one of the $\mathcal{B}_i$'s outputs 1 only if one of the elements x_0^i or x_1^i is in $Y(\Lambda)$, which implies that at least one of the γ_b^i is statistically hidden, which in turn implies that γ_b is statistically hidden. The receiver's security follows from the fact that for every i and for $\Lambda \leftarrow I_n$, it is hard to distinguish between $x_0 \in_R L(\Lambda)$ and $\mathcal{A}_i(\Lambda, x_0)$.

The subset membership problem M. Our subset membership problem $\mathbf{M} = \{I_n\}_{n \in \mathbb{N}}$ is based on the one defined in [CS02]. However, we incorporate here several modifications.

1. *Problem samplability.* For every n, I_n is a samplable distribution that generates an instance description Λ as follows: On input 1^n,
   (a) Generate two random n bit safe primes p, q; namely, primes p and q such that $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ are odd primes. Let $N = pq$ and $T \triangleq 2^{\lceil \log N \rceil}$.
   (b) Choose a random element $\mu \in_R \mathbb{Z}_N^*$, and output $\Lambda = (N, \mu)$, which specifies (X, W, R) in the following way: $X \triangleq J_N$, $L \triangleq \langle \mu^T \rangle$ is the subgroup generated by $\mu^T (\text{mod } N)$, $W \triangleq \{0, 1, \ldots, \lfloor N/4 \rfloor\}$, and $R \triangleq \{(\mu^{T_r}, r) : r \in W\}$.

   Notice that $L \subseteq \{x \in J_N : order(x) \text{ is odd}\}$, for every (even maliciously chosen) N. This is the case since the order of μ divides $\phi(N)$ (which is the order of Z_N^*), and 2 divides $\phi(N)$ at most $\lceil \log N \rceil$ times. Thus, 2 divides the order of μ at most $\lceil \log N \rceil$ times. This implies that the order of $\mu^T (\text{mod } N)$ (where $T = 2^{\lceil \log N \rceil}$) is co-prime to 2, and thus is odd.

   Moreover, for every honestly chosen N, with overwhelming probability $L = QR_N = \{x \in J_N : order(x) \text{ is odd}\}$. This follows from the fact that QR_N is a cyclic group of order N', which implies that a random element in QR_N generates QR_N with overwhelming probability. Moreover, since the order of every element in QR_N is co-prime to 2, it follows that $\langle \mu^T \rangle = \langle \mu^2 \rangle$.

For every $\Lambda = (N, \mu)$, let $Y(\Lambda) = J_N \backslash QR_N$. Then for every (even maliciously chosen) Λ, it holds that $Y(\Lambda) \subseteq \{x \in J_N : order(x) \text{ is even}\}$.

2. *Member samplability:* On input $\Lambda = (N, \mu)$, choose a random $r \in W$, and output $\mu^{Tr} (\text{mod } N)$ together with its corresponding witness r.
3. *Non-member samplability* $\mathcal{A}_i$*:* On input $\Lambda = (N, \mu)$ and $x \in X(\Lambda)$, if $i \in J_N$ then $\mathcal{A}_i(\Lambda, x)$ outputs the element $i \cdot x (\text{mod } N)$. If $i \notin J_N$ then $\mathcal{A}_i(\Lambda, x)$ outputs x.[16]
4. Y*-Verifiability* $\mathcal{B}_i$*:* On input (Λ, x_0, x_1), if $i \in J_N$, then $\mathcal{B}_i(\Lambda, x_0, x_1)$ outputs 1 when both $x_0, x_1 \in J_N$ and $x_b / x_{b-1} = i (\text{mod } N)$ for some $b \in \{0, 1\}$. If $i \notin J_N$ then $\mathcal{B}_i(\Lambda, x_0, x_1)$ always outputs 1.

We would like to prove that $\mathbf{M}$ is a Y-verifiably samplable hard subset membership problem. The hardness of $\mathbf{M}$ follows from the fact that with overwhelming probability over $\Lambda = (N, \mu) \leftarrow I_n$, it holds that $L(\Lambda) = QR_N$. In order to prove that $\mathbf{M}$ is Y-verifiably samplable, we need to prove that $\mathbf{M}$ satisfies the following three properties: member samplablility, non-member samplability, and Y-verifiability. It is easy to see that the member samplability property holds. In order to see that the non-member samplability property holds it suffices to notice that under the Quadratic Residuosity Assumption, for every large enough n, for $\Lambda = (N, \mu) \leftarrow I_n$, and for every $i = 1, \ldots, \log^3 N$, it is hard to distinguish between $x \in_R L(\Lambda)$ and $x' = \mathcal{A}_i(\Lambda, x)$. In order to show that the Y-verifiability property holds, it suffices to show that the Y-verifiability property holds for a single i. This we show under the Extended Riemann Hypothesis, using the following well known result from algebraic number theory.

Lemma 1 ([BS96], 8.5.9). *Assume the Extended Riemann Hypothesis. Let H be a non-trivial subgroup of Z_N^* of index d, and let C be a coset of H. Then the least prime whose residue belongs to C is $O(d^2 \log^2 N)$.*

Assume the Extended Riemann Hypothesis. We first use Lemma 1 to prove that for every (maliciously chosen) N one of the elements in $\{1, ..., \log^3 N\} \cap J_N$ is also an element in $J_N \setminus QR_N$.[17]

Consider any $N = p_1^{a_1} ... p_k^{a_k}$. Let G be the subgroup of Z_N^* consisting of all elements which are squares modulo p_1. Let $H \triangleq J_N \cap G$. Notice that both G and J_N are subgroups of Z_N^* of index 2, and that H is a subgroup of Z_N^* of index 4. Now let g be any element in J_N which is *not* a square modulo p_1 (i.e., $g \in J_N \setminus G$), and let $C = gH$ be a coset of H. According to Lemma 1, the Extended Riemann Hypothesis implies that one of the elements in $\{1, 2, \ldots, x\}$, where $x = O(d^2 \log^2 N) < \log^3 N$, must be an element in C. Notice that all elements in C are non-squares modulo p_1, which implies that $C \subseteq J_N \setminus QR_N$.

[16] For $i \notin J_N$, x can be distinguished from $i \cdot x$, since it is easy to check whether an element in $\mathbb{Z}_N^*$ has Jacobi symbol 1. Thus, in this case we simply let $\mathcal{A}_i(\Lambda, x)$ output x, to make sure that it is hard to distinguish x from $\mathcal{A}_i(\Lambda, x)$.

[17] In what follows we use our assumption that N is not a power of a single prime (if N is a power of a single prime then $J_N \setminus QR_N = \emptyset$).

Thus, we conclude that one of the elements in $\{1, 2, \ldots, x\} \subset \{1, 2, \ldots, \log^3 N\}$ is in $J_N \setminus QR_N$.

Fix any (even maliciously chosen) $\Lambda = (N, \mu)$, and let $i \in \{1, ..., \log^3 N\} \cap J_N$ be an element in $J_N \setminus QR_N$. It is easy to see that for every x_0, x_1, if both are not in $Y(\Lambda) = J_N \setminus QR_N$ then $\mathcal{B}_i(\Lambda, x_0, x_1)$ outputs 0. Moreover, for any x_0, x_1, such that $x_b \in L(\Lambda)$ and $x_{1-b} \in \mathcal{A}_i(\Lambda, x_b)$ (for some b), it holds that $x_{1-b} = i \cdot x_b$ and $x_0, x_1 \in J_N$ (since $i \in J_N$), and thus $\mathcal{B}_i(\Lambda, x_0, x_1)$ outputs 1.

$(\frac{1}{2}, Y)$-universal projective hash family for $\mathbf{M}$. Consider the projective hash family $(\mathcal{H}, K, S, \alpha, G)$, defined as follows. For every $\Lambda = (N, \mu) \in \mathbf{M}$:

- Let $K(\Lambda) = \{0, 1, \ldots, \lfloor \frac{N}{2} \rfloor\}$ and let $K = \bigcup_{\Lambda \in \mathbf{M}_i} K(\Lambda)$.
- Let $G(\Lambda) = J_N$ and let $G = \bigcup_{\Lambda \in \mathbf{M}_i} G(\Lambda)$.
- For every $k \in K(\Lambda)$, let $H_k(x) = x^k (\text{mod } N)$.
- For every $k \in K(\Lambda)$, let $\alpha(k) = \mu^{Tk} (\text{mod } N)$, where $T \triangleq 2^{\lceil \log N \rceil}$.

Claim. The hash family $(\mathcal{H}, K, S, \alpha, G)$ is an efficient $(\frac{1}{2}, Y)$-universal projective hash family for $\mathbf{M}$.

Proof. It is straightforward to verify that all efficiency requirements hold. As for the projection requirement, it follows easily from the fact for every $k \in K(\Lambda)$, and for every $x \in L(\Lambda)$:

$$H_k(x) = x^k (\text{mod } N) = (\mu^{Tr})^k (\text{mod } N) = (\mu^{Tk})^r (\text{mod } N) = \alpha(k)^r (\text{mod } N)$$

We next prove that this projective hash family is $(\frac{1}{2}, Y)$-universal. Fix any (even maliciously chosen) $\Lambda = (N, \mu)$, and fix any $x \in Y(\Lambda) = J_N \setminus QR_N$. As was previously mentioned, x is of even order. Let $Z \triangleq \phi(N)/GCD(\phi(N), T)$. Note that Z is an odd number, and that $\mu^{TZ} = 1 (\text{mod } N)$. Also note that for every s there are (at least) two distinct elements $k, k + Z \in K(\Lambda)$ such that $s = \alpha(k) = \alpha(k + Z)$. Thus, in order to prove that the $(\frac{1}{2}, Y)$-universal property holds, it remains to prove that $x^Z \neq 1 (\text{mod } N)$, which follows immediately from the fact that x is of even order.

Acknowledgements. First and foremost I would like to thank Alon Rosen. Although he refused to be a co-author of this paper, Alon's comments, suggestions and involvement played an essential part in the creation of this work. I would like to thank Ronald Cramer for pointing out and explaining the notion of smooth projective hashing. I would like to thank Shien Jin Ong for pointing out a crucial mistake that I had in the initial stage of this work. I would like to thank Eric Bach for pointing out Lemma 1. Finally, I would like to thank Shai Halevi and Shafi Goldwasser for their great comments and simplifying suggestions.

References

[AIR01] William Aiello, Yuval Ishai, Omer Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In *EUROCRYPT 2001*, pages 119-135, 2001.

[BS96] E.Bach and J. Shallit. Algorithmic Number Theory, Vol. 1: Efficient Algorithms. MIT Press, 1996.

[BM89] M. Bellare and S. Micali. Non-Interactive Oblivious Transfer and Applications. In *CRYPTO '89*, pages-547-557, 1989.

[CS98] R. Cramer and V.Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO 1998*, pages 13-25, 1998.

[CS02] R. Cramer and V.Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Eurocrypt 2002*, Springer-Verlag (LNCS 2332), pages 45-64, 2002.

[Cre87] C. Crépeau. Equivalence between two flavours of oblivious transfers. In *CRYPTO 1987*, pages 350-354, 1987.

[CCM98] C. Cachin, C. Crépeau, Julien Marcil. Oblivious Transfer with a Memory-Bounded Receiver. In *FOCS 1998*, pages 493-502, 1998.

[DHRS04] Y. Z. Ding, D. Harnik, A. Rosen, R. Shaltiel. Constant-Round Oblivious Transfer in the Bounded Storage Model. In *TCC 2004*, pages 446-472,2004.

[Fr98] John B. Fraleigh. A first course in abstract algebra , 7th edition, Addison-Wesley 1998.

[EGL85] S. Even and O. Goldreich and A. Lempel. A Randomized Protocol for Signing Contracts. In *Communications of the ACM 28:6*, pages 637–647, 1985.

[GL03] R. Gennaro and Y. Lindell. A Framework for Password-Based Authenticated Key Exchange. In *EUROCRYPT 2003*, pages 524-543, 2003.

[Gol04] O. Goldreich. Foundations of Cryptography - Volume 2 (Basic Applications). Cambridge University Press, 2004.

[GMW87] O. Goldreich and S. Micali and A. Wigderson. How to Play any Mental Game - A completeness Theorem for Protocols with Honest Majority. In *STOC 1987*, pages 218-229, 1987.

[Hai04] Iftach Haitner. Implementing Oblivious Transfer Using Collection of Dense Trapdoor Permutations. In *TCC 2004*, pages 394-409, 2004.

[IR89] R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In *STOC 89*, pages 44–61, 1989.

[KOY01] J. Katz, R. Ostrovsky, M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In *EUROCRYPT 2001*, pages 475-494, 2001.

[Kil88] J. Kilian. Founding Cryptography on Oblivious Transfer. 20th ACM Symposium on the Theory of Computing, pages 20–31, 1988.

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA 2001*, pages 448-457,2001

[Pa99] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT 1999*, pages 223-238, 1999.

[Rab81] M. O. Rabin. How to Exchange Secrets by Oblivious Transfer. TR-81, Harvard, 1981.

[Y86] A. C. Yao. How to Generate and Exchange Secrets. In *FOCS 1986*, pages 162-167, 1986.

On Robust Combiners for Oblivious Transfer and Other Primitives

Danny Harnik[1,⋆], Joe Kilian[2], Moni Naor[1,⋆], Omer Reingold[1,†], and Alon Rosen[3]

[1] Dept. of Computer Science and Applied Math., Weizmann Institute of Science
{danny.harnik, moni.naor, omer.reingold}@weizmann.ac.il
[2] Yianilos Labs
joe@pnylab.com
[3] CSAIL, MIT
alon@csail.mit.edu

Abstract. A (1,2)-robust combiner for a cryptographic primitive $\mathcal{P}$ is a construction that takes two candidate schemes for $\mathcal{P}$ and combines them into one scheme that securely implement $\mathcal{P}$ even if one of the candidates fails. Robust combiners are a useful tool for ensuring better security in applied cryptography, and also a handy tool for constructing cryptographic protocols. For example, we discuss using robust combiners for obtaining universal schemes for cryptographic primitives (a universal scheme is an explicit construction that implements $\mathcal{P}$ under the sole assumption that $\mathcal{P}$ exists).

In this paper we study what primitives admit robust combiners. In addition to known and very simple combiners for one-way functions and equivalent primitives, we show robust combiners for protocols in the world of public key cryptography, namely for Key Agreement(KA).

The main point we make is that things are not as nice for Oblivious Transfer (OT) and in general for secure computation. We prove that there are no "transparent black-box" robust combiners for OT, giving an indication to the difficulty of finding combiners for OT. On the positive side we show a black box construction of a $(2, 3)$-robust combiner for OT, as well as a generic construction of $(1, n)$-robust OT-combiners from any $(1, 2)$-robust OT-combiner.

At the mouth of two witnesses ... shall the matter be established
Deuteronomy Chapter 19.

1 Introduction

Not putting all your eggs in one basket is commonly considered good advice and this should be no different in cryptography. Suppose that we have a two cryptographic schemes that we generally trust to be secure for some task. It makes

⋆ Research supported in part by a grant from the Israel Science Foundation.
† Research supported by US-Israel Binational Science Foundation Grant 2002246.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 96–113, 2005.

a lot of sense to try and combine these two into one scheme that is guaranteed to be secure even in case that one of the two original schemes was broken. For example, we have several encryption schemes that are based on various unproven number theoretic assumptions, such as the hardness of factoring or of computing discrete logarithms. We would like to combine these into one encryption scheme that is secure if at least one of these unproven assumptions happens to be true. We call such a construction a *Robust Combiner*.[1] This is a scheme that combines two different schemes and is robust to the failure of just one of them.

Definition 1.1 ((k,n)-Robust Combiner (Informal)). *A (k,n)-Robust Combiner for a cryptographic primitive $\mathcal{P}$ is a construction that takes n candidate schemes for $\mathcal{P}$ and combines them into one scheme such that if at least k of the candidates indeed implement $\mathcal{P}$ then the combiner also implements $\mathcal{P}$.*

In general, the most interesting combiners are $(1,2)$-robust combiners as they are essential and at times sufficient for constructing $(1,n)$-robust combiners (n is some parameter, typically related to the security parameter). For ease of notations we will sometimes write just robust combiner or simply combiner when we actually mean a $(1,2)$-robust combiner.

Robust combiners are by all means not new in cryptography. Several practical constructions try to combine several primitives to achieve stronger security guarantees. For example, Asmuth and Blakely [1] suggest a method of combining two encryption schemes of which only one can be trusted. Another example is the widely used idea of repeatedly encrypting a message several times with different keys in order to enhance security, an idea that dates back as far as Shannon and found in many applications since. This relates to combiners as security holds in the case that the integrity of some of the keys is compromised, but at least one remains secure. Also, Herzberg [15] discusses the notion of combiners explicitly (see the related work section, Section 1.2).

There are plenty of other practical motivations for combiners, we briefly give a few: For example, using software from a few sources that are not entirely trusted (e.g., when running an election and using electronic ballots from a few vendors). Combiners can also be used to avoid bugs in software, rather than checking the correctness of a software (as in [5]), combine several different versions, hoping that at least one is correct. One can also consider physical sources used for cryptography (e.g. noisy channels) that cannot necessarily be trusted.

From the point of view of theoretical cryptography robust combiners are also valuable. Combiners are useful tools in constructions and reductions between cryptographic primitives. This happens in scenarios where it is guaranteed that one of several constructions exist. We give two examples:

- Levin [21] (see exposition in [13]) introduced a *Universal-one way funcion (OWF)* which is an explicit construction that is guaranteed to be a OWF under the sole assumptions that one-way functions exist at all. The property

[1] This notion is called a *Tolerant Construction* in [15].

of one-way functions that allows for this universal constructions is the fact that they admit robust combiners.
- In the construction of pseudo-random generators (PRG) from OWFs by Hastad et al. [14] a polynomial number of candidates for PRG are given, one of which is guaranteed to be a PRG. These are then combined (the combiner is a simple XOR of the output) into one PRG construction.

1.1 Our Contributions

In this paper we study what cryptographic primitives have or don't have robust combiners. We start by showing that simple robust combiners exist for OWF (this is common knowledge) and its equivalents (such as private key encryption, pseudo-random generators, functions and permutations, digital signatures and bit commitment). We then present a robust combiner for Key Agreement (KA) and, similarly, Public Key Encryption (PKE).

On Robust Combiners for Oblivious transfer: The abundance and relative simplicity of robust combiners may lead to the belief that all primitives have simple combiners. However, this is not the case for the fundamental oblivious transfer primitive (OT) and thus for any non trivial task of secure computation. We define the notion of black-box combiners, giving several refinements to this notion. Our main result shows the following:

Theorem 1.2 *(informal) There exists no "transparent black-box" construction of a robust OT-combiner.*

Transparent black-box combiners are black-box combiners with a specific property. In general, it is required that every time a party calls one of the candidates, then the other party learns about this call (all messages generated by the candidate are actually sent to the other party).

Theorem 1.2 can be viewed as an indication of the hardness of the problem of constructing combiners for OT. The point being that most of the known examples of combiners are transparent black-box combiners. More precisely, this indicates that achieving a combiner for OT will likely use the OT protocol outside of its context (and perhaps not as an interactive process).

A good example and an exception to the generally simple combiners is the combiner for bit commitments. This combiner uses the commitment candidates in a non interactive manner in order to generate a OWF. It then uses the HILL reduction [14] together with [22] to build a commitment from a OWF. Such a strategy seems hard for OT since there are black box separations of OT from simpler and less structured primitives such as OWFs and KA [18, 12].

Positive results for OT: On a more positive note, we show a very efficient black box construction of a (2,3)-robust OT-combiner. We also point out that it is easy to construct an OT protocol based on the assumption that at least one of the assumptions regarding factoring or the discrete logarithms is correct. This is because there are known constructions of OT from such assumptions (and in general from any trapdoor permutation [11]) that have perfect (and guaranteed) security for the receiver, in which case constructing combiners is simple.

(1,n)-robust combiners and universal schemes: We discuss the notion of a universal scheme for a cryptographic primitive (following Levin's [21] universal OWF) and show that primitives that admit (1,n)-robust combiners also have universal schemes. We then study cases where (1,2)-combiners are sufficient for (1,n)-combiners. Among others, it is shown that a (1,2)-robust combiner for OT also gives a construction of a universal scheme for OT (the construction makes use of the efficient (2,3)-robust combiner for OT shown here).

Other points: A delicate point when discussing combiners for a primitive $\mathcal{P}$ is the question of functionality. In some settings, while one of the input candidates is guaranteed to be secure, the other one is not even guaranteed to have the functionality of $\mathcal{P}$, making things more involved. In general, one way to overcome this is by first testing the functionality of a possibly faulty candidate. For instance, the combiner for KA first constructs a KA where the two parties agree only with reasonably high probability, and then reduces the probability of disagreement to a negligible one using an error correcting code.

1.2 Related Work

As mentioned before, robust combiners have already been used and studied. In particular the work of Herzberg [15] focuses on robust combiners in cryptography. This work puts more emphasis on efficiency and specifically the use of the parallel and cascade constructions as combiners and shows combiners for various primitives including OWF, signatures, MACs and others.

Implicit use of combiners is abundant. For example, the idea of using multiple encryptions is widely used in practice. This practice is in fact advocated in the NESSIE consortium recommendations [23]. Also the TLS (Transport Layer Security) specification [17] combines two hash functions (SHA1 and MD5) to give better assurance of security. We quote from [17]: " In order to make the PRF as secure as possible, it uses two hash algorithms in a way which should guarantee its security if either algorithm remains secure."[2] Lately Dodis and Katz [10] studied the use of multiple encryptions with respect to CCA2 security, giving a robust combiner for CCA2 secure encryption schemes using signatures. Hohenberger and Lysyanskaya [16] discuss how to securely combine two potentially insecure software implementations. Another related concept is given in Brickell and McCurley [6] and Shoup [25] that show schemes that achieve two different types of security based on two different number theoretic assumptions.

The work of Damgard, Kilian and Salvail [9] is somewhat relevant to the OT-combiner. This work discusses a weak version of OT called (p, q)-OT that has probability p of compromising the sender's security and probability q of compromising the receiver's. It is shown that a fully secure OT can be constructed from a (p, q)-OT if and only if $p + q < 1$. In our setting where two candidates

[2] Note that the concatenation of hash functions as suggested in the TLS [17] is indeed a combiner in the sense that it is guaranteed to be as secure as the candidate that remains secure. This does not however guarantee an increase of the security in case that candidates are secure, as was shown by Joux [19].

for OT are given, one can obtain a (p, q)-OT with $p = q = \frac{1}{2}$ simply by choosing one of the candidates at random. Therefore, the impossibility result of [9] for $p + q \geq 1$ gives some intuition for the impossibility of OT-combiners. However the result for $p + q \geq 1$ relies heavily on the fact that the errors p and q are assumed to be uncorrelated events, which is not the case in the setting of combiners. On the other hand, for (2,3)-robust combiners, we can get an OT protocol with $p = q = \frac{1}{3}$ and use the reduction from [9] (although the (2,3)-robust OT-combiner presented here is much more efficient, a property that is used in Section 5.1).

2 Notations and Definitions

We denote by *PPTM* a probabilistic polynomial time Turing machine. In general, our definitions view adversaries as uniform machines, though all results in this paper also apply for definitions of security against non-uniform adversaries. An Oracle PPTM is a PPTM that also has access to one or more oracles.

2.1 Cryptographic Primitives

The notion of a cryptographic primitive ranges from basic non-interactive constructs such as one-way functions, digital signatures and encryption to more "high-level" interactive protocols such as secret key exchange and oblivious transfer. Due to lack of space and the difficulty of actually giving a complete definition to this notion, we refrain from presenting a full definition of a primitive, and only highlight the key issues (see [24] for a formal definition).

In principle, the definition of a primitive $\mathcal{P}$ includes a description of the *functionality* of the primitive (computable in polynomial time), along with a definition of *security*. The functionality defines what the primitive should do, whereas the security deals with the ability of an adversary of a certain class (e.g., all PPTMs) to learn something from an *implementation* of the functionality. This ability is captured by a relation between possible machines (modelling the adversary) and functions (modelling the implementation). The relation defines when a machine *breaks* an implementation. For an implementation to be secure, it is required that no machine in the class of adversaries can break this implementation.

In the special case of interactive primitives, the functionality of the primitive can be divided into two parts: (1) The *next message* function M, which determines the next message to be sent by a party (given its partial view of the interaction). (2) An *output function* O, which determines a party's local output (given the view of the entire interaction). A protocol is then obtained by letting each of the sides alternately generate their next message by applying the function M to their own local inputs, randomness and partial view (up to that point in the interaction). At the end of interaction each side feeds its view to the function O to get its local output.

2.2 Robust Combiners

Combiners receive as input candidates for implementing a primitive $\mathcal{P}$. In principle, the candidates can be either given as the code of a PPTM, or via an oracle

that implements it. The basic definition of a combiner does not take this issue into consideration and admits any kind of usage of the candidate implementations.

Definition 2.1 ((k,n)-Robust Combiner). *Let $\mathcal{P}$ be a cryptographic primitive. A* (k,n)-Robust Combiner *for $\mathcal{P}$ is a PPTM that gets n candidate schemes as inputs, and implements $\mathcal{P}$ while satisfying the following two properties:*

1. *If at least k candidates securely implement $\mathcal{P}$ then the combiner also securely implements $\mathcal{P}$.*
2. *The running time of the combiner is polynomial in the security parameter m, in n and in the lengths of the inputs to $\mathcal{P}$.*[3]

Note that in general a combiner could completely ignore the candidate implementations and implement $\mathcal{P}$ directly. However, we are interested in combiners whose security relies on the security guarantees of the candidates. It thus makes sense to consider a more restrictive notion of a combiner, in which both the construction and its proof are conducted in a "black-box" manner.

Definition 2.2 (Black-Box Combiner). *A (1,2)-robust combiner is said to be* **black-box** *if the following conditions hold:*

1. Black-box implementation: *The combiner is an oracle PPTM given access to the candidates via oracle calls to their implementation function.*
2. Black-box proof: *For every candidate there exists an oracle PPTM R^A (with access to A) such if adversary A breaks the combiner, then the oracle PPTM R^A breaks the candidate.*[4]

In the case of interactive primitives several additional restrictions on the usage of the underlying candidate implementations make sense. One natural restriction that comes into mind is to require that the combiner totally ignores the implementation and simply relies on the functionality and security of one of the candidates (e.g., the combiner for KA presented in Section 3.3).

Definition 2.3. *A* **third party black-box combiner** *is a black-box combiner where the candidates behave like trusted third parties. The candidates give no transcript to the players, but rather take their inputs and return outputs.*

In some situations the above notion is too restrictive and a transcript is actually needed for enabling the construction of a combiner (for example, constructing a OWF cannot be done from a third party implementation for OT). In this paper we also discuss a relaxation of third party black-box combiners, that allows access to the transcripts of the protocols as well.

[3] Here we make the implicit assumption that the candidates themselves run in polynomial time. See a further discussion in Section 3.1.

[4] In the case of (k,n)-robust combiners then there are at least $n - k + 1$ candidates that can be broken in this manner.

Definition 2.4. *A* **transparent black-box combiner** *is a black-box combiner for an interactive primitive, where every call to a candidate's next message function M is followed by this message being sent to the other party.*

This notion can be thought of as allowing the use of the primitive only in the context of the protocol (rather than allowing free off-line use of its oracles). Note that the notion of black-box combiners (considered in Definition 2.2) is less restrictive than the third party and transparent ones. A black-box combiner is given unlimited off-line access to the oracles that generate the protocol whereas the other combiners are not. Note that in the case of non-interactive primitives the three notions defined above are equivalent.

3 Positive Results

3.1 The General Framework for Robust Combiners

Cryptographic primitives are mainly about security. So naturally the emphasis when constructing robust combiners will be that these primitives indeed remain secure in face of the unfortunate case that one of the candidates actually breaches security. However, there are some subtleties that need to be discussed. In some settings, hardly anything is known about the candidates at hand other than the fact that one of them is good. Specifically, only one candidate is guaranteed to have the intended functionality. For example, a faulty candidate for a OWF, may not only be easy to invert, but might also be hard to compute in the easy direction (computing the function might be impossible for all PPTM). Other primitives might have additional functionalities (other than running time) that should be taken into consideration. For example, in the KA (key agreement) both parties should output the same key (the agreement). In this section we present approaches for dealing with these issues, dealing separately with running time and other functionalities.

Running time: In general, one cannot expect to be able to check that a candidate for a cryptographic primitive always halts in polynomial time unless the specific polynomial bound on the running time is known in advance. We therefore assume that the polynomial bound is given as input to the scheme. For example, a robust OWF-combiner gets as input a polynomial $p(\cdot)$ and the security parameter 1^m along with the two candidates f_A, f_B. Now, when a combiner invokes a candidate, it allows it to run for at most $p(m)$ steps, and if it does not halt then the output of the candidate is set to some fixed value (e.g. to the all zero string).[5]

Functionality Test: A possible approach for testing the functionality of a candidate (such as agreement in key agreement or the transfer of the chosen secret in oblivious transfer) is presented. This method may sometimes be helpful but at other times impossible, depending on the specific primitive at hand. The idea is

[5] Unless relevant, we omit the parameter $p(\cdot)$ from the text and simply assume that the running time of all candidates is polynomial (a fact that is essential for most proofs of security).

to have each party simulate n^2 random off-line executions of the candidate, and accept only if the candidate always satisfies its defined functionality. For example, in key agreement, each party simulates a random execution by playing the roles of both players and checking whether they agree. After passing the test we are assured that with probability $1 - O(2^{-n})$ the candidate does what it is supposed to with probability at least $1 - \frac{1}{n}$. While this is a rather weak guarantee, it is sometimes sufficient (as in the case of KA-combiners, see Section 3.3).

Note: The functionality and time tests may not be always necessary. For example, when trying to combine two constructions based on two different computational assumptions, the functionality and running time are usually guaranteed by the design of these constructions. These tests are necessary however in the general case where nothing is known (e.g., in universal schemes, see Section 5.1).

3.2 Robust Combiners for OWFs and Equivalents

It is has long been known that one-way functions (OWF) have simple robust combiners. For example, as pointed out in [15], simple concatenation of the OWF candidates on independent inputs suffices . More precisely, given candidates f_A and f_B, let $F(x, y) = f_A(x)|f_B(y)$ (where f_A and f_B run in polynomial time).

Lemma 3.1 *F is a robust OWF-combiner.*

Lemma 3.1 (proof omitted) implies that all the primitives that are known to be *equivalent* to OWF have robust combiners. By equivalent we mean, primitives that have reductions to and from OWFs. Some of the more noteworthy equivalent primitives are semantically secure private key encryption, pseudo-random generators, functions and permutations, digital signatures and bit commitments. The combiners for these primitives follow since given two candidates for primitive $\mathcal{P}$ (from the list above), one can use the reduction from OWF to $\mathcal{P}$ to create two candidates for OWFs. These two are then combined using the OWF-combiner, which in turn is used to construct the primitive $\mathcal{P}$ from a OWF (with the opposite reduction from $\mathcal{P}$ to OWFs).

Note, however, that for most of these primitives going via the reductions to and from OWF is an overkill, and much more efficient and direct combiners can be found. For example a combiner for pseudo-random generator is simply one that XORs the outputs (thus the heavy reduction of [14] from pseudo-random generators to OWFs may be avoided). An exception is the case of bit commitments for which we are only aware of the combiner via the OWF. Unlike the non-interactive primitives in the list (that have very simple combiners), the suggested combiner for commitment is highly inefficient (this issue is further discussed in Section 6).

3.3 Robust Key Agreement Combiner

Theorem 3.2 *There exists a robust KA-combiner. The combiner reaches agreement with all but a negligible probability. Furthermore, its round complexity is at most that of the candidate with the higher number of rounds.*[6]

[6] By round complexity we mean the worst-case round complexity.

Observe that a KA-combiner can be easily achieved if the functionality of both candidates is guaranteed. The KA-combiner simply outputs an XOR of the outputs in the two candidates. If the functionality is not guaranteed, then the combiner for KA is constructed in two stages. First a KA-combiner with *relaxed agreement* is constructed (a protocol in which the parties agree with all but a polynomially small fraction). Then this is turned into a KA were the agreement happens with overwhelming probability using an error correction code.

We note that the KA-combiner is a third party BB combiner. Also, since a 2 message KA protocol is equivalent to semantically secure (against chosen plaintext attacks) Public Key Encryption (PKE), and since the KA-combiner maintains the same round complexity, we also get for free a robust PKE-combiner.

4 On Robust Combiners for Oblivious Transfer

4.1 Impossibility of Black Box Robust OT-Combiner

In contrast to all the other primitives mentioned here that had robust combiners (and usually very simple ones), the situation of OT is left open. We do not know of any OT-combiner, simple or complicated. The main result in this section indicates that this is indeed a much harder problem.

We start by giving some intuition: Suppose that a combiner does exist for OT, then this combiner works for every two candidates that we plug in, as long as one of them is actually secure. The idea is to show that the OT-combiner will work just as well when given two faulty candidates where one candidates is secure only for Alice while the other is secure only for Bob. But this immediately yields a contradiction, since two such faulty candidates can be naively constructed under no assumptions at all, giving rise to an OT protocol based on no hardness assumptions, which is impossible. An actual proof of this idea shows that any attack on the combined OT taking the two faulty candidates, can be translated to an attack on the combined OT that takes one truly secure candidate (and one faulty candidate), thus breaking the security of the combiner. This intuition is formalized in the following theorem:

Theorem 4.1 *There exists no construction of a transparent black-box robust OT-combiner.*

We note that it is simpler to show the impossibility for third party BB combiners. However, we work a bit harder in order to capture the notion of transparent BB combiners, and in particular combiners that can also use the transcript of the protocol. Recall that a transparent black-box combiner (defined in Section 2) is one in which the candidates are given via a "next message" oracle and an output oracle. Whenever one of the parties calls a next message oracle it is required to send the message generated to the other party.

Proof: Similarly to many black box impossibility results (starting with the seminal paper of Impagliazzo and Rudich [18]), Theorem 4.1 is proved by trying to show a "world" in which OT exists, but OT-combiners do not. The argument

however must be changed, since in every world that has OT, an OT-combiner does exists, simply by running the correct OT protocol. Instead, the actual proof shows two worlds such that every transparent black-box OT-combiner is insecure in at least one of them (we show this even in the semi-honest model[7]).

We define two oracle worlds: World1 and World2. Both worlds contain a PSPACE-complete oracle and an implementation of two OTs: OT_A and OT_B. The implementation is rather straightforward and each OT is composed of three oracles (presented below). In each world *one* of the implementations is *made* flawed by adding an inverter for some of the oracles. Specifically, in World1 OT_A is insecure and OT_B is secure and in World2 OT_A is secure and OT_B is insecure. We now consider the application of the combiner on candidates OT_A and OT_B in these two worlds. Let us denote the resulting protocol by OT_{cmb}. Note that OT_A and OT_B look identical from the point of view of the combiner in both worlds. Since in each of the worlds one of the OTs is secure, then by the definition of the combiner, OT_{cmb} should be secure in both worlds. We claim that OT_{cmb} fails in at least one of these worlds, thus contradicting the existence of a combiner.

To prove our claim, we appeal to a "bare" world containing solely a PSPACE-complete oracle (this oracle already exists in World1 and World2 and we will explain its significance shortly). In the bare world we simulate OT_{cmb}. Note that OT_{cmb} is well defined once we plug in an implementation for OT_A and OT_B. Therefore, in order to implement OT_{cmb} we give a *naive* implementation of both OT_A and OT_B in the bare world. For this the sender (of OT_{cmb}) simulates OT_A and the receiver (of OT_{cmb}) simulates OT_B. Meaning for example that whenever OT_{cmb} requires the receiver (of OT_{cmb}) to query one of the functions of OT_A (either as a receiver or as a sender of this invocation of OT_A), the receiver will ask the sender (of OT_{cmb}) this query (in the clear) and the sender will return the answer (again in the clear). These simulations of OT_A and OT_B are obviously insecure and therefore the resulting implementation of OT_{cmb} is also be insecure (in fact, no implementation of OT can be secure in the bare world since with the PSPACE oracle no crypto is possible).

So what is the point of considering this naive implementation of OT_{cmb} in a world where this implementation is bound to fail? The point is that the failure of OT_{cmb} in the bare world translates to a failure of OT_{cmb} either in World1 or in World2. This is exactly what we need to complete the proof. Assume for example that the receiver of OT_{cmb} in the bare world learns both secrets. In this case, the receiver of OT_{cmb} in World2 can also learn both secrets. This is because the receiver of in the bare world gains precisely the same knowledge as the receiver of in World2: Both learn all inputs to OT_B. In the bare world the receiver learns it as it simulates OT_B and in World2 the receiver learns it through the inverter for OT_B. We next give a formal proof.

We present an oracle that enables the execution of an OT protocol. This oracle is composed of a triplet of functions $OT = (f_1, f_2, R)$ as follows:

[7] Recall that in the *Semi-Honest* model the parties follow the protocol as prescribed, but perhaps later try to learn more information than intended.

- f_1 is a length tripling random function[8] that takes the receiver's choice bit c and randomness r_R and outputs $m_1 = f_1(r_R, c)$ that is used as the receiver's message.
- f_2 is also a length tripling random function that takes the sender's inputs s_0, s_1 and randomness r_S and the receiver's message m_1 and outputs the sender's message $m_2 = f_2(r_S, s_0, s_1, m_1)$.
- R is called by the receiver, it takes m_2 along with r_R and c and outputs the secret s_c (if the inputs are consistent).

Using the above oracle it is possible to implement a secure OT protocol in a straightforward manner. Notice that the receiver learns the secret of his choice. On the other hand since the parties cannot invert the random functions, then the messages give them essentially no additional information. Moreover, this is true even in the presence of a PSPACE-complete oracle as stated in the following claim (given here without a proof):

Claim 4.2 *The procedure defined by the oracle (f_1, f_2, R) is a secure OT protocol even in the presence of a PSPACE-complete oracle.*

In addition to the functions enabling an OT oracle, we may add another oracle for breaking such an OT. This oracle simply inverts the functions f_1, f_2, and thus leaks both secrets to the receiver and the choice bit to the sender.[9]

The two worlds: We can now define the two oracle worlds.

- **World1**, contains:
  1. A PSPACE-complete oracle.
  2. Two OT oracles $OT_A = (f_1^A, f_2^A, R^A)$ and $OT_B = (f_1^B, f_2^B, R^B)$.
  3. The oracle Inv_A for inverting OT_A.
- **World2**, contains:
  1. A PSPACE-complete oracle.
  2. Two OT oracles $OT_A = (f_1^A, f_2^A, R^A)$ and $OT_B = (f_1^B, f_2^B, R^B)$.
  3. The oracle Inv_B for inverting OT_B.

Now consider a robust OT-combiner that takes OT_A and OT_B as candidates and call this protocol OT_{cmb}. By the definition of a combiner, OT_{cmb} should securely implement an OT protocol in each of the two worlds, since in both worlds one of the two candidates remains secure. We achieve a contradiction by showing that if the OT-combiner is transparent black-box then there exists an attack on the protocol OT_{cmb} in at least one of the two worlds.

[8] A length tripling random function is a function $f : \{0,1\}^n \to \{0,1\}^{3n}$ that sends each input value to an independently chosen random value in the output domain.

[9] This inverting oracle is possible since with overwhelming probability f_1 and f_2 are one-to-one functions (as they are random function and by a simple birthday argument are not likely to have any collisions).

The Bare World and Simulating OT_{cmb}: To show the attack on OT_{cmb}, we turn to the "bare" world that contains just the PSPACE oracle but not the OT oracles. For every instantiation of OT_{cmb} in worlds 1 and world 2, we give a matching protocol called OT_{bare} in the bare world. The new protocol in the bare world imitates OT_{cmb} with the exception that the sender of OT_{cmb} simulates the oracle OT_A (we explain below what we mean by simulating an OT oracle) and the receiver of OT_{cmb} simulates OT_B. Note that the sender of OT_{cmb} simulates OT_A whether he acts as sender or receiver in the specific invocation of OT_A (and likewise for the receiver of OT_{cmb} simulating OT_B).

A party simulates an oracle by answering every query to the functions f_1 or f_2 by a random value. In addition, the party records all the answers he gave to queries during the protocol's execution. When the function R of the OT oracle is queried, the party simply inverts the functions using the records he stored in memory, allowing him to reply with the proper answer.[10]

The first thing to notice is that OT_{bare} indeed has the functionality of an OT protocol (perhaps up to a negligible error). This is since the simulations of OT_A and OT_B are consistent with actual OT implementations. On the other hand, OT_{bare} cannot be a secure OT protocol. This is simply due to the known fact that there exists no unconditional construction for OT (this may be traced back to [7] or even [4]). We give a more precise interpretation of this claim: An OT protocol is defined by the parties inputs s_0, s_1 and c, along with their respective random coins r_S and r_R. Denote by $view_S^{OT}$ (and $view_R^{OT}$) the view of the sender (receiver) in this protocol (including the party's input, randomness and the messages in the transcript).

Claim 4.3 *For every implementation of OT, there exist poly-time procedures A_S and A_R with access to the PSPACE-complete oracle such that for every choice of s_0, s_1, c, r_S, r_R we have that either $A_S(view_S^{OT}) = c$ or $A_R(view_R^{OT}) = (s_0, s_1)$.*

In particular, there exist two procedures A_S and A_R as above that constitute a break of OT_{bare}. Claim 4.3 is given here without a proof.

The attack on OT_{cmb}: To conclude the proof, we show that the attack A_S on OT_{bare} can be equally successful when applied in World1 on OT_{cmb}. Likewise, the attack A_R, can be used on OT_{cmb} in World2.

The attack of the sender of OT_{cmb} in World1 is achieved as follows: Let the sender simulate the view of the sender in OT_{bare}, and run A_S on this view. Denote the simulated view by $view_S^{World1}$, which is generated as follows: The sender runs OT_{cmb} as prescribed (recall that OT_{bare} follows the same prescription), but whenever the oracle OT_A is called (by either side), the sender calls the inverting oracle Inv_A and records the inputs and outputs to the oracle. Here it is crucial that the sender is aware of all the answers that the receiver got for his queries to OT_A, which is guaranteed by the transparent black-box structure of the combiner.

[10] We assume here that the OT oracle answers a $\perp$ whenever an illegal input is given. The simulator simply does the same when he gets a query with an input that was not previously in his memory (and thus not a legal input).

The way OT_{bare} was constructed ensures that every choice of oracles OT_A and OT_B is consistent with some randomness of the sender and receiver in OT_{bare}. Thus for every execution of OT_{cmb} with inputs s_0, s_1 and c, there exists an execution of OT_{bare} with the same inputs, for which $view_S^{World1}$ is identical to the view in OT_{bare} (denoted $view_S^{bare}$). Thus whenever $A_S(view_S^{bare}) = c$ in the bare world, then is also $A_S(view_S^{World1}) = c$ in World1. Respectively, in World2, for the exact same execution of OT_{cmb}, the receiver can simulate the view in the same corresponding execution of OT_{bare}. Now whenever $A_R(view_R^{bare}) = (s_1, s_2)$ in the bare world, then is also $A_R(view_R^{World2}) = (s_1, s_2)$ in World2. Combining this with Claim 4.3 we get that there exist procedures A'_S and A'_R, such that for every execution of OT_{cmb}, either A'_S breaks it in World1 or A'_R breaks it in World2. □

4.2 (2,3)-Robust OT-Combiner

The results of the previous section indicate that (1,2)-Robust OT-combiners seem out of our reach at this point. We can however give a solution to the slightly more modest task of (2,3)-Robust OT-combiner. This solution is a third party black-box combiner and relies on some often used techniques of Crépeau and Kilian [8] for amplifying the security in weak versions of OT protocols.

Claim 4.4 *There exists a (2,3)-robust OT-combiner scheme.*

Furthermore, the (2,3)-combiner is very efficient, making just 6 calls to the candidates. The efficiency is essential for the application Section 5. Due to space limitations we give here only a description of the construction and defer the proof of its security to the full version of this paper. For simplicity we will discuss OT on single bits, although everything can be generalized for strings in a straightforward manner.

Consider 3 candidates for oblivious transfer OT_A, OT_B, OT_C. We first use a construction that takes 2 OT candidates and always maintains the security of the receiver.
$R(OT_A, OT_B)(s_0, s_1; c)$ is defined as follows:

1. The sender chooses a random bit r
2. The receiver chooses random bits c_0, c_1 such that $c_0 \oplus c_1 = c$
3. The parties run $OT_A(r, r \oplus s_0 \oplus s_1; c_0)$ and $OT_B(r \oplus s_0, r \oplus s_1; c_1)$
4. The receiver outputs the XOR of his outputs in both executions.

We next present another construction that takes 3 candidates for OT and strongly protects the sender. Define $S(OT_A, OT_B, OT_C)(s_0, s_1; c)$ as follows:

1. The sender chooses random bits r_0^A, r_0^B, r_0^C and r_1^A, r_1^B, r_1^C subject to $r_0^A \oplus r_0^B \oplus r_0^C = s_0$ and $r_1^A \oplus r_1^B \oplus r_1^C = s_1$.
2. The parties run $OT_A(r_0^A, r_1^A; c)$, $OT_B(r_0^B, r_1^B; c)$ and $OT_C(r_0^C, r_1^C; c)$.
3. The receiver outputs the XOR of his outputs in the three candidates.

Finally, define $OT_{AB} = R(OT_A, OT_B)$, $OT_{AC} = R(OT_A, OT_C)$ and $OT_{BC} = R(OT_B, OT_C)$. The (2,3)-robust OT-combiner is defined as $S(OT_{AB}, OT_{AC}, OT_{BC})$.

An alternative construction is to create an OT that is secure with probability $\frac{2}{3}$ simply by first randomly choosing one of the three candidates and then applying it. In [9] it was shown how such an OT can be amplified to one that is secure with all but a negligible probability. However the construction presented here is much more efficient, a fact that is later used in Section 5.

5 From (1,2)-Combiners to (1,n)-Combiners

(1,2)-robust combiners are essential for the existence of (1,n)-robust combiners. It is interesting to study under what conditions (1,2)-combiners suffice for the construction of (1,n)-combiners.

For some primitives, (1,k)-combiners can be reached as a simple extension of the construction of (1,2)-combiners (for instance, the KA-combiner presented in Section 3.3 extends easily). However, this is not clear for all combiners, and depends on the specific primitive at hand. We try to give more generic answers to the question posed above.

The natural construction takes the k candidates and organizes them as leaves of a binary tree, and applies the (1,2)-Robust $\mathcal{P}$-combiner scheme for every internal node (in a bottom up fashion). Now, by the properties of the combiner, for every node that securely implements $\mathcal{P}$, its ancestor must also securely implement $\mathcal{P}$. The output of the whole tree must therefore also securely implement $\mathcal{P}$ since the root is an ancestor to all leaves. This construction is indeed a (1,k)-combiner provided that the running time is polynomial. However, the depth of the tree is logarithmic in k, and if the running time of the (1,2)-combiner is m times that of its candidates, then the running time of the whole construction is $m^{\Omega(logk)}$. Thus, in order for the running time to be polynomial, m must be a constant. We distinguish between general (polynomial time) combiners and very efficient ones. A combiner is said to be **very efficient** if its running time is bounded by a constant times the running time of its candidates (for example, the combiners for OWFs and pseudorandom generators are very efficient).

Lemma 5.1 *For any $\mathcal{P}$ and for all k, any very efficient $(1,2)$-Robust $\mathcal{P}$-combiner can be turned into a $(1,k)$-Robust $\mathcal{P}$-combiner.*

As suggested above, the tree construction is not efficient when the running time of the (1,2)-combiner is polynomial time. This is troubling since if a (non-BB) OT-combiner is eventually found, it is not very likely that it will be a very efficient one. Nevertheless, it will still suffice for constructing (1,n)-combiners for OT. We show that given a very efficient (2,3)-combiner, one can construct (1,n)-combiners from any (not necessarily very efficient) (1,2)-combiner. This result along with the very efficient (2,3)-combiner for OT (Section 4.2) allow us to focus our attention on constructing (1,2)-combiners for OT.

Theorem 5.2 *Any (1,2)-robust combiner for OT, can be used to construct a $(1,k)$-Robust combiner for OT.*

Proof: The construction of the (1,k)-combiner makes use of the (2,3)-robust OT-combiner presented in Section 4.2. The crux being that the (2,3)-combiner for OT is very efficient (in fact it makes just 6 calls to its candidates, though we simply use the multiplicative constant c). Divide the k candidates into three groups of size $\frac{2}{3}k$ such that each candidate appears in at least two of the groups. For instance, take the first two thirds as group 1, the second two thirds as group 2 and the first and last thirds as group 3. The construction recursively computes a $(1, \frac{2}{3}k)$-combiner on each of these groups. The 3 outcomes of these combiners are given as input to the (2,3)-combiner.

Since one candidate is guaranteed to be secure, at least 2 of the combiners on the 3 groups implement secure OT protocols. Therefore the outcome of the (2,3)-combiner securely implements OT. Let $t(k)$ be the running time of the (1,k)-combiner. The base of the recursion is a (1,2)-combiner that takes a polynomial time (say $t(2) = n^d$ for constant d). The recursion gives us running time $t(k) = 3c \cdot t(\frac{2k}{3})$. Altogether this gives $t(k) = (3c)^{\log_{3/2} k} \cdot n^d$ which is polynomial.

Note that the $(1, n)$-combiner can be made to work even if the OT functionality[11] of the candidates is not guaranteed. This is achieved by testing the functionality of all candidates in advance and using an error correcting code as well. □

5.1 Universal Schemes for Primitives

Definition 5.3 (Universal Schemes). *A universal scheme $\mathcal{U}$ for a cryptographic primitive $\mathcal{P}$ is an explicit construction with the property that if the primitive $\mathcal{P}$ exists, then $\mathcal{U}$ is a secure implementation of $\mathcal{P}$.*

Levin [21] introduced such a scheme for OWFs. He showed an explicit function which is a OWF under the sole assumption that OWFs exist. In a sense, the meaning of such a universal scheme $\mathcal{U}$ for $\mathcal{P}$ is that any proof of existence for $\mathcal{P}$ is guaranteed to be a constructive one, since, once $\mathcal{P}$ is proved to exist then $\mathcal{U}$ is an explicit implementation of $\mathcal{P}$. The property that allowed Levin's universal-OWF schemes is the existence of robust combiners for OWFs. We try to formalize this connection for other primitives as well.

Lemma 5.4 *For any cryptographic primitive $\mathcal{P}$, a Universal-$\mathcal{P}$ scheme can be provided if:*

1. *There is a known polynomial $p(\cdot)$ such that if there exists an implementation for $\mathcal{P}$ then there also exists an an implementation for $\mathcal{P}$ with running time bounded by $p(n)$.*
2. *$\mathcal{P}$ admits $(1, k)$-robust combiners (for k a super-constant ($\omega(1)$) in the security parameter n).*

Proof: The general idea of the universal scheme is to go over all possible implementation programs, hoping that at least one of them will fulfill our need. Then use the combiner to unite all of the programs into one that implements

[11] The OT functionality is that the receiver gets the bit of his choice.

the primitive $\mathcal{P}$. More precisely, the universal scheme $\mathcal{U}$ with security parameter 1^n goes over all of the Turing machines[12] of description length at most $\log n$ and unites them into one program using the (1,n)-Robust $\mathcal{P}$-combiner with polynomial $p(n)$ as a time bound. So if a program implementing $\mathcal{P}$ exists then for some large enough n, this program is included in the n programs that $\mathcal{U}$ executes, and by the robustness of the $\mathcal{P}$-combiner we have that $\mathcal{U}$ is also an implementation of $\mathcal{P}$. □

Lemma 5.4 requires two properties of a primitive, the first asks that a time bound will be known on some implementation of $\mathcal{P}$. This property is very likely to be true about cryptographic primitives due to a **padding argument** similar to the one used for universal OWF in [13] (omitted her due to space limitations). The padding argument works for most of the primitives we can think of. However care needs to be taken with primitives such as pseudorandom generators where padding of the input must also involve padding of the output. In the case of pseudorandom generators, for instance, it is easy to find a slightly modified argument that will work.

As corollaries of the above claims we get explicit constructions of many cryptographic primitives such as Universal-OWF and Universal-KA. Due to Theorem 5.2 We further get:

Corollary 5.5 *Any (1,2)-robust combiner for OT, can be used to construct a universal-OT scheme.*

Note that in a computational setting, a (1,2)-combiner for OT can simply ignore the candidate and run a universal-OT scheme (this is a non-black-box combiner). Thus, in this setting we can say that (1,2)-combiners for OT exist if and only if universal schemes for OT exist.

6 Open Problems

The most intriguing question that rises from this paper is whether robust OT-combiners exist or not. Black box impossibility results have already been bypassed in the past, for instance, in the work of Barak [2]. We believe however, that solving this problem will require an altogether new technique. The techniques of [2, 3] do not seem to help here. The reason being that this technique makes use of an explicit description of the *adversary's* program, which is of importance when dealing with malicious behavior. However our problem is interesting also in the semi-honest model, where such a program is constant. Another direction would be to try and reach a full impossibility result for general (rather then transparent) black-box combiners.

An interesting question about combiners regards the bit commitment primitive. For computationally hiding and statistically binding bit commitments we

[12] This step depends highly on the nature of the primitive $\mathcal{P}$. For example, if $\mathcal{P}$ is an interactive protocol (like key agreement), then we enumerate interactive Turing machines.

know how to build robust combiners, via the reduction to OWFs (given a OWF, commitments can be constructed using the reductions of Naor [22] and Hastad et al. [14]) which gives an inefficient combiner. It would be interesting to find a direct and more efficient combiner for commitments. For statistically hiding (computationally Binding) commitments the question of combiners is altogether open. [13] It is worth noting that no third party BB combiners for commitments exist (for both types of commitments). This can be shown using the same technique from our impossibility result for OT (Theorem 4.1). On the positive side, there is a very efficient (2,3)-robust combiner for commitments (shown in [15]). Also, if the security of one of the party's is guaranteed then constructing combiners for commitments is easy. An example for such a case is commitments to strings where the commitment is much shorter than the secret (as in [20]).

Acknowledgements. We thank the anonymous referees for their helpful comments.

References

1. C.A. Asmuth and G.R. Blakely. An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. *Computers and Mthematics and Applications*, 7:447–450, 1981.
2. B. Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115, 2001.
3. B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *43rd FOCS*, pages 345–355, 2002.
4. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th STOC*, 1988.
5. M. Blum and S. Kannan. Designing programs that check their work. In *21st ACM Symposium on the Theory of Computing*, pages 86–97, 1989.
6. E. Brickell and K. McCurley. An interactive identification scheme based on discrete logarithms and factoring. *Journal of Cryptology*, 5(1):29–39, 1992.
7. B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. *SIAM Journal on Disc. Math.*, 4(1):36–47, 1991.
8. C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *29th FOCS*, pages 42–52, 1988.
9. I. Damgard, J. Kilian, and L. Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Eurocrypt '99*, pages 56–73, 1999.
10. Y. Dodis and J. Katz. Chosen ciphertext security of multiple encryption. In *TCC 05*, pages 188–209, 2005.
11. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
12. Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335, 2000.

[13] A reduction of statistically hiding commitments to OWFs would suffice for constructing combiners, however, at this point such a reduction is not known.

13. O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.
14. J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 29(4):1364–1396, 1999.
15. A. Herzberg. On tolerant cryptographic constructions. ECCC, TR02-135, 2002.
16. S. Hohenberger and A. Lysyanskaya. How to securely outsource cryptographic computations. In *TCC 05*, pages 264–282, 2005.
17. IETF. The tls protocol, version 1.1. www.ietf.org, 2002.
18. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61, 1989.
19. A. Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *CRYPTO '04*, volume 3152, pages 306–316. Springer.
20. J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *24th STOC*, pages 723–732, 1992.
21. L. A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7:357–363, 1987.
22. M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.
23. Nessie. Recommended cryptographic primitives. www.cryptonessie.org, 2003.
24. O. Reingold, L. Trevisan, and S. Vadhan. Notions of reducibility between cryptographic primitives. In *TCC '04*, pages 1–20, 2004.
25. V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *Advances in Cryptology – EUROCRYPT ' 2000*, volume 1807, pages 275–288, 2000.

Efficient Identity-Based Encryption Without Random Oracles

Brent Waters

Stanford University
bwaters@cs.stanford.edu

Abstract. We present the first efficient Identity-Based Encryption (IBE) scheme that is fully secure without random oracles. We first present our IBE construction and reduce the security of our scheme to the decisional Bilinear Diffie-Hellman (BDH) problem. Additionally, we show that our techniques can be used to build a new signature scheme that is secure under the computational Diffie-Hellman assumption without random oracles.

1 Introduction

Identity-Based Encryption allows for a party to encrypt a message using the recipient's identity as a public key. The ability to use identities as public keys avoids the need to distribute public key certificates. This can be very useful in applications such as email where the recipient is often off-line and unable to present a public-key certificate while the sender encrypts a message.

The first efficient and secure method for Identity-Based Encryption was put forth by Boneh and Franklin [4]. They proposed a solution using efficiently computable bilinear maps that was shown to be secure in the random oracle model. Since then, there have been schemes shown to be secure without random oracles, but in a weaker model of security know as the Selective-ID model [9, 1]. Most recently, Boneh and Boyen [2] described a scheme that was proved to be fully secure without random oracles; the possibility of such a scheme was to that point an open problem. However, their scheme is too inefficient to be of practical use.

We present the first efficient Identity-Based Encryption scheme that is fully secure without random oracles. The proof of our scheme makes use of an algebraic method first used by Boneh and Boyen [1] and the security of our scheme reduces to the decisional Bilinear Diffie-Hellman (BDH) assumption.

We additionally show that our IBE scheme implies a secure signature scheme under the computational Diffie-Hellman assumption without random oracles. Previous practical signature schemes that were secure in the standard model relied on the Strong-RSA assumption [12, 11] or the Strong-BDH assumption [3].

1.1 Related Work

Shamir [16] first presented the idea of Identity-Based Encryption as a challenge to the research community. However, the first secure and efficient scheme of

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 114–127, 2005.

Boneh and Franklin[4] did not appear until much later. The authors took a novel approach in using efficiently computable bilinear maps in order to achieve their result.

Canetti et. al. [9] describe a weaker model of security for Identity-Based Encryption that they term the *Selective-ID* model. In the Selective-ID model the adversary must first declare which identity it wishes to be challenged on before the global parameters are generated. The authors provide a scheme that is provably secure in the Selective-ID model without random oracles. Boneh and Boyen [1] improve upon this result by describing an efficient scheme that is secure in the Selective-ID model.

Finally, Boneh and Boyen [2] describe a scheme that is fully secure without random oracles. However, their construction is too inefficient to be of practical use.

1.2 Organization

We organize the rest of the paper as follows. In Section 2 we give our security definition. In Section 3 we describe our complexity assumptions. In Section 4 we present the construction of our IBE scheme and follow with a proof of security in Section 5. In Section 6 we discuss how our scheme can be extended to a hierarchical identity-based encryption scheme and how that can be used to achieve CCA-security. We discuss the transformation to a signature scheme in Section 7. Finally, we conclude in Section 8.

2 Security Definitions

In this section we present the definition of semantic security against passive adversaries for Identity-Based Encryption. This definition was first described by Boneh and Franklin [4]. Consider the following game played by an adversary. The game has four distinct phases:

Setup. The challenger generates the master public parameters and gives them to the adversary.

Phase 1. The adversary is allowed to make a query for a private key, v, where v is an identity specified by the adversary. The adversary can repeat this multiple times for different identities.

Challenge. The adversary submits a public key, v^*, and two messages M_0 and M_1. The adversary's choice of v^* is restricted to the identities that he did not request a private key for in Phase 1. The challenger flips a fair binary coin,γ, and returns an encryption of M_γ under the public key v^*.

Phase 2. Phase 1 is repeated with the restriction that the adversary cannot request the private key for v^*.

Guess. The adversary submits a guess, γ', of γ.

Definition 1 (IBE Semantic Security). *An Identity-Based Encryption scheme is (t, q, ϵ)-semantically secure if all t-time adversaries making at most q private key queries have at most an ϵ in breaking our scheme.*

3 Complexity Assumptions

We briefly review the facts about groups with efficiently computable bilinear maps. We refer the reader to previous literature [4] for more details.

Let $\mathbb{G}, \mathbb{G}_1$ be s groups of prime order p and g be a generator of $\mathbb{G}_1$. We say $\mathbb{G}_1$ has an admissible bilinear map, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$, into $\mathbb{G}_1$ if the following two conditions hold. The map is bilinear; for all a, b we have $e(g^a, g^b) = e(g, g)^{ab}$. The map is non-degenerate; we must have that $e(g, g) \neq 1$.

3.1 Decisional Bilinear Diffie-Hellman (BDH) Assumption

The challenger chooses $a, b, c, z \in \mathbb{Z}_p$ at random and then flips a fair binary coin β. If $\beta = 1$ it outputs the tuple $(g, A = g^a, B = g^b, C = g^c, Z = e(g, g)^{abc})$. Otherwise, if $\beta = 0$, the challenger outputs the tuple $(g, A = g^a, B = g^b, C = g^c, Z = e(g, g)^z)$. The adversary must then output a guess β' of β.

An adversary, $\mathcal{B}$, has at least an ϵ advantage in solving the decisional BDH problem if

$$\Big| \Pr\left[\mathcal{B}\left(g, g^a, g^b, g^c, e(g, g)^{abc}\right) = 1\right] - \Pr\left[\mathcal{B}\left(g, g, g^a, g^b, g^c, e(g, g)^z\right) = 1\right] \Big| \geq 2\epsilon$$

where the probability is over the randomly chosen a, b, c, z and the random bits consumed by $\mathcal{B}$. We refer to the left hand side as $\mathcal{P}_{BDH}$ and the right hand side as $\mathcal{R}_{BDH}$.

Definition 2. *The decisional (t, ϵ)-BDH assumption holds if no t-time adversary has at least ϵ advantage in solving the above game.*

3.2 Computational Diffie-Hellman (DH) Assumption

The challenger chooses $a, b \in \mathbb{Z}_p$ at random and outputs $(g, A = g^a, B = g^b)$. The adversary then attempts to output $g^{ab} \in \mathbb{G}$. An adversary, $\mathcal{B}$, has at least an ϵ advantage if

$$Pr\left[\mathcal{B}\left(g, g^a, g^b\right) = g^{ab}\right] \geq \epsilon$$

where the probability is over the randomly chosen a, b and the random bits consumed by $\mathcal{B}$.

Definition 3. *The computational (t, ϵ)-DH assumption holds if no t-time adversary has at least ϵ advantage in solving the above game.*

4 Construction

Our construction can be viewed as a modification of the Boneh-Boyen [1] scheme. We first present our construction then describe its relation to the Boneh-Boyen scheme.

Let $\mathbb{G}$ be a group of prime order, p, for which there exists an efficiently computable bilinear map into $\mathbb{G}_1$. Additionally, let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ denote the bilinear map and g be the corresponding generator. The size of the group is determined by the security parameter. Identities will be represented as bitstrings of length n, a separate parameter unrelated to p. We can also let identities be bitstrings of arbitrary length and n be the output length of a collision-resistant hash function, $H : \{0,1\}^* \rightarrow \{0,1\}^n$. Our construction follows.

Setup. The system parameters are generated as follows. A secret $\alpha \in \mathbb{Z}_p$ is chosen at random. We choose a random generator, $g \in \mathbb{G}$, and set the value $g_1 = g^\alpha$ and choose g_2 randomly in $\mathbb{G}$. Additionally, the authority chooses a random value $u' \in \mathbb{G}$ and a random n-length vector $U = (u_i)$, whose elements are chosen at random from $\mathbb{G}$. The published public parameters are g,g_1, g_2,u', and U. The master secret is g_2^α.

Key Generation. Let v be an n bit string representing an identity, v_i denote the *ith* bit of v, and $\mathcal{V} \subseteq \{1, \ldots, n\}$ be the set of all i for which $v_i = 1$. (That is $\mathcal{V}$ is the set of indicies for which the bitstring v is set to 1.) A private key for identity v is generated as follows. First, a random $r \in \mathbb{Z}_p$ is chosen. Then the private key is constructed as:

$$d_v = \left(g_2^\alpha \left(u' \prod_{i \in \mathcal{V}} u_i \right)^r, g^r \right).$$

Encryption. A message $M \in \mathbb{G}_1$ is encrypted for an identity v as follows. A value $t \in \mathbb{Z}_p$ is chosen at random. The ciphertext is then constructed as

$$C = \left(e(g_1, g_2)^t M, g^t, \left(u' \prod_{i \in \mathcal{V}} u_i \right)^t \right).$$

Decryption. Let $C = (C_1, C_2, C_3)$ be a valid encryption of M under the identity v. Then C can be decrypted by $d_v = (d_1, d_2)$ as:

$$\begin{aligned} C_1 \frac{e(d_2, C_3)}{e(d_1, C_2)} &= \left(e(g_1, g_2)^t M\right) \frac{e(g^r, \left(u' \prod_{i \in \mathcal{V}} u_i\right)^t)}{e(g_2^\alpha \left(u' \prod_{i \in \mathcal{V}} u_i\right)^r, g^t)} \\ &= \left(e(g_1, g_2)^t M\right) \frac{e(g, \left(u' \prod_{i \in \mathcal{V}} u_i\right)^{rt})}{e(g_1, g_2)^t e(\left(u' \prod_{i \in \mathcal{V}} u_i\right)^{rt}, g)} = M. \end{aligned}$$

4.1 Efficiency

If the value of $e(g_1, g_2)$ is cached then encryption requires on average $\frac{n}{2}$ (and at most n) group operations in $\mathbb{G}$, two exponentiations in $\mathbb{G}$, one exponentiation in $\mathbb{G}_1$, and one group operation in $\mathbb{G}_1$. Decryption requires two bilinear map computations, one group operation in $\mathbb{G}_1$ and one inversion in $\mathbb{G}_1$.

4.2 Similarity to Boneh-Boyen

Our construction is a modification of Boneh and Boyen's [1] in that that for an identity v we evaluate $u' \prod_{i \in \mathcal{V}} u_i$ whereas in their scheme they evaluate $u' g_1^v$, where v is interpreted as an integer. (We technically are referring to the first scheme presented in Boneh-Boyen [1] when only a level one hierarchy is used. Although, our scheme can be extended to be a hierarchical scheme in an analogous manner.) In the next section we show that, remarkably, this small modification is all that is needed to make the scheme fully secure.

5 Security

We now prove the security of our scheme.

Theorem 1. *Our IBE- scheme is* (t, q, ϵ) *secure assuming the decisional* $(t + O(\epsilon^{-2} \ln(\epsilon^{-1}) \lambda^{-1} \ln(\lambda^{-1})), \frac{\epsilon}{32(n+1)q})$ *BDH assumption holds, where* $\lambda = \frac{1}{8(n+1)q}$.

Proof. Suppose there exists a (t, q, ϵ)-adversary, $\mathcal{A}$ against our scheme. We construct a simulator, $\mathcal{B}$, to play the decisional BDH game. The simulator will take BDH challenge $(g, A = g^a, B = g^b, C = g^c, Z)$ and outputs a guess, β', as to whether the challenge is a BDH tuple. The simulator runs $\mathcal{A}$ executing the following steps.

5.1 Simulator Description

Setup. The simulator first sets an integer, $m = 4q$, and chooses an integer, k, uniformly at random between 0 and n. It then chooses a random n-length vector, $\overrightarrow{x} = (x_i)$, where the elements of $\overrightarrow{x}$ are chosen uniformly at random from the integers between 0 and $m-1$ and a value, x', chosen uniformly at random between 0 and $m-1$. Let X^* denote the pair $(x', \overrightarrow{x})$ Additionally, the simulator chooses a random $y' \in \mathbb{Z}_p$ and an n-length vector, $\overrightarrow{y} = (y_i)$, where the elements of $\overrightarrow{y}$ are chosen at random in $\mathbb{Z}_p$. These values are all kept internal to the simulator.

Again, for an identity v we will let $\mathcal{V} \subseteq \{1, \ldots, n\}$ be the set of all i for which $v_i = 1$ For ease of analysis we define three functions. We define $F(v) = (p - mk) + x' + \sum_{i \in \mathcal{V}} x_i$ and define $J(v) = y' + \sum_{i \in \mathcal{V}} y_i$. Finally, we define a binary function $K(v)$ as

$$K(v) = \begin{cases} 0, & \text{if } x' + \sum_{i \in \mathcal{V}} x_i \equiv 0 \pmod{m} \\ 1, & \text{otherwise.} \end{cases}$$

The simulator assigns $g_1 = A$ and $g_2 = B$. It then assigns the public parameters $u' = g_2^{p-km+x'} g^{y'}$ and U as $u_i = g_2^{x_i} g^{y_i}$. From the perspective of the adversary the distribution of the public parameters is identical to the real construction.

Phase 1. The adversary, $\mathcal{A}$, will issue private key queries. Suppose the adversary issues a query for an identity v. If $K(v) = 0$ the simulator aborts and randomly chooses its guess β' of the challenger's value β.

Otherwise, the simulator chooses a random $r \in \mathbb{Z}_p$. Using the technique described by Boneh and Boyen [1] it constructs the private key, d, as

$$d = (d_0, d_1) = \left(g_1^{\frac{-J(v)}{F(v)}} (u' \prod_{i \in \mathcal{V}} u_i)^r, g_1^{\frac{-1}{F(v)}} g^r \right).$$

Let $\tilde{r} = r - \frac{a}{F(v)}$. Then we have

$$\begin{aligned} d_0 &= g_1^{\frac{-J(v)}{F(v)}} (u' \textstyle\prod_{i \in v} u_i)^r \\ &= g_1^{\frac{-J(v)}{F(v)}} (g_2^{F(v)} g^{J(v)})^r \\ &= g_2^a (g_2^{F(v)} g^{J(v)})^{-\frac{a}{F(v)}} (g_2^{F(v)} g^{J(v)})^r \\ &= g_2^a (u' \textstyle\prod_{i \in \mathcal{V}} u_i)^{r - \frac{a}{F(v)}} \\ &= g_2^a (u' \textstyle\prod_{i \in \mathcal{V}} u_i)^{\tilde{r}}. \end{aligned}$$

Additionally, we have

$$d_1 = g_1^{\frac{-1}{F(v)}} g^r = g^{r - \frac{a}{F(v)}} = g^{\tilde{r}}.$$

This simulator will be able to perform this computation iff $F(v) \neq 0 \mod p$. For ease of analysis the simulator will only continue (not abort) in the sufficient condition where $K(v) \neq 0$. (If we have $K(v) \neq 0$ this implies $F(v) \neq 0 \mod p$ since we can assume $p > nm$ for any reasonable values of p, n, and m).

Challenge. The adversary next will submit two messages $M_0, M_1 \in \mathbb{G}_1$ and an identity, v^*. If $x' + \sum_{i \in \mathcal{V}^*} x_i \neq km$ the simulator will abort and submit a random guess for β'. Otherwise, we have $F(v^*) \equiv 0 \pmod p$ and the simulator will flip a fair coin, γ, and construct the ciphertext

$$T = (ZM_\gamma, C, C^{J(v^*)}).$$

Suppose that the simulator was given a BDH tuple, that is $Z = e(g, g)^{abc}$. Then we have

$$T = \left(e(g, g)^{abc} M_\gamma, g^c, g^{cJ(v^*)} \right) = \left(e(g_1, g_2)^c M_\gamma, g^c, (u' \prod_{i \in \mathcal{V}^*} u_i)^c \right).$$

We see that T is a valid encryption of M_γ.

Otherwise, we have that Z is a random element of $\mathbb{G}$. In that case the ciphertext will give no information about the simulator's choice of γ.

Phase 2. The simulator repeats the same method it used in Phase 1.

Guess. Finally, the adversary $\mathcal{A}$ outputs a guess γ' of γ.

Artificial Abort. At this point the simulator is still unable to use the output from the adversary. An adversary's probability of success could be correlated with the probability that the simulator needs to abort. This stems from the fact that two different sets of q private key queries may cause the simulator to abort with different probabilities. In the worst case we might worry that $\Pr[\gamma = \gamma' | \overline{\texttt{abort}}] - \frac{1}{2} = 0$ (or some negligible value) in the simulation even if $\Pr[\gamma = \gamma'] - \frac{1}{2} = \epsilon$ for some non-negligible ϵ.

The simulator corrects for this by forcing all possible sets of queries of the adversary to cause the simulator to abort with (almost) the same probability (1-λ), where (1-λ) is a lower bound on any set of private key queries causing an abort before this stage.

Let $\overrightarrow{v} = v_1, \ldots v_q$ denote the private key queries made in Phase 1 and Phase 2 and let v^* denote the challenge identity and we let $\mathcal{V}^* \subseteq \{1, \ldots, n\}$ be the set of all i for which $v_i^* = 1$. (All of these values are defined at this point in the simulation.) First, we define the function $\tau(X', \overrightarrow{v}, v^*)$, where X' is a set of simulation values $x', x_1, \ldots, x_n$, as

$$\tau(X', \overrightarrow{v}, v^*) = \begin{cases} 0, & \text{if } (\bigwedge_{i=1}^{q} K(v_i) = 1) \wedge x' + \sum_{i \in \mathcal{V}^*} x_i = km \\ 1, & \text{otherwise.} \end{cases}$$

The function $\tau(X', \overrightarrow{v}, v^*)$ will evaluate to 0 if the private key and challenge queries $\overrightarrow{v}, v^*$ will not cause an abort for a given choice of simulation values, X'. We can now consider the probability over the simulation values for a given set of queries, $\overrightarrow{v}, v^*$, as $\eta = \Pr_{X'}[\tau(X', \overrightarrow{v}, v^*) = 0]$.

The simulator samples $O(\epsilon^{-2} \ln(\epsilon^{-1}) \lambda^{-1} \ln(\lambda^{-1}))$ times the probability η by choosing a random X' and evaluating $\tau(X', \overrightarrow{v}, v^*)$ to compute an estimate η'. We emphasize that the sampling does not involve running the adversary again. Let $\lambda = \frac{1}{8nq}$, be the lower bound on the probability of not aborting for any set of queries. (We show how to calculate λ below.) Then if $\eta' \geq \lambda$ the simulator will abort with probability $\frac{\eta' - \lambda}{\eta'}$ (not abort with probability $\frac{\lambda}{\eta'}$) and take a random guess β'. Otherwise, the simulator will not abort.

If the simulator has not aborted at this point it will take check to see if the adversary's guess, $\gamma' = \gamma$. If $\gamma' = \gamma$ then the simulator outputs a guess $\beta' = 1$, otherwise it outputs $\beta = 0$.

This concludes the description of the simulator.

5.2 Analysis

Our simulator is difficult to analyze directly since it might abort before all of the queries are made. For ease of exposition we now describe a second simulation, which we will use to reason about the output distribution of the first simulation.

Setup. The simulator chooses the secret key g_2^{α} as in the construction and then chooses $X^*, \overrightarrow{y}$ as in the first simulation and derives u', U in the same way. It then runs the adversary.

Phase 1. The simulator responds to private key queries by using the master key as in the construction, in this way all queries can be answered.

Challenge. The simulator receives the challenge messages M_0, M_1. The second simulator will flip two coins β and γ. If $\beta = 0$ then it encrypts a random message and if $\beta = 1$ it encrypts M_γ.

Phase 2. Same as Phase 1.

Guess. The simulator receives a guess γ' from the adversary. At this point the simulator has seen as the private key queries and the challenge query $(\overrightarrow{v}, v^*)$. It evaluates the function $\tau(X^*, \overrightarrow{v}, v^*)$ and aborts if it evaluates to 1, outputting a random guess of β'.

Artificial Abort. The last step is done in exactly the same way as the first simulation. This ends the description.

We first equate the probabilities of the both simulators with the following claim.

Claim. The probabilities $\Pr[\beta' = \beta]$ are the same in both the first and second simulations we described.

Proof. The second simulation runs the adversary completely and receives all of its queries. In the guess phase it checks if $\tau(X^*, \overrightarrow{v}, v^*) = 1$ and aborts if so. The check decides if there was a point where the first simulator would have needed to abort during the simulator and take a random guess. If so the second simulator aborts and takes a random guess itself. Additionally, all public parameters, private key queries, and challenge ciphertexts have the same distribution up to the point of a possible abortion. The artificial abort stages are also identical. Therefore, we can reason that the output distributions will be the same. □

For purpose of exposition, we will now derive the success of the simulator in terms of the second simulator. However, due to Claim 5.2 the discussion applies to both simulators equally.

Claim. The probability of the simulation not aborting by the guess phase is at least $\lambda = \frac{1}{8(n+1)q}$.

Proof. We calculate a lower bound, λ as the lower bound of $\Pr_{X'}[\tau(X', \overrightarrow{v}, v^*) = 0]$ for all $\overrightarrow{v}, v^*$. Without loss of generality we can assume the adversary always makes the maximum number of queries, q. For any set of q queries $v_1, \ldots, v_q$ and challenge identity, v^*, we have $\Pr[\overline{\texttt{abort}}] = \Pr[(\bigwedge_{i=1}^{q} K(v_i) = 1) \wedge \sum_{i \in \mathcal{V}^*} x_i = km]$. We can then lower bound the probability of not aborting as follows.

$$\Pr[(\bigwedge_{i=1}^{q} K(v_i) = 1) \wedge \sum_{i \in \mathcal{V}^*} x_i = km] \tag{1a}$$

$$= (1 - \Pr[\bigvee_{i=1}^{q} K(v_i) = 0]) \Pr[\sum_{i \in \mathcal{V}^*} x_i = km | \bigwedge_{i=1}^{q} K(v_i) = 1] \tag{1b}$$

$$\geq (1 - \sum_{i=1}^{q} \Pr[K(v_i) = 0]) \Pr[\sum_{i \in \mathcal{V}^*} x_i = km | \bigwedge_{i=1}^{q} K(v_i) = 1] \tag{1c}$$

$$= (1 - \frac{q}{m}) \Pr[\sum_{i \in \mathcal{V}^*} x_i = km | \bigwedge_{i=1}^{q} K(v_i) = 1] \tag{1d}$$

$$= \frac{1}{n+1}(1 - \frac{q}{m}) \Pr[K(v^*) = 0 | \bigwedge_{i=1}^{q} K(v_i) = 1] \tag{1e}$$

$$= \frac{1}{n+1}(1 - \frac{q}{m}) \frac{\Pr[K(v^*) = 0]}{\Pr[\bigwedge_{i=1}^{q} K(v_i) = 1]} \Pr[\bigwedge_{i=1}^{q} K(v_i) = 1 | K(v^*) = 0] \tag{1f}$$

$$\geq \frac{1}{(n+1)m}(1 - \frac{q}{m}) \Pr[\bigwedge_{i=1}^{q} K(v_i) = 1 | K(v^*) = 0] \tag{1g}$$

$$= \frac{1}{(n+1)m}(1 - \frac{q}{m})(1 - \Pr[\bigvee_{i=1}^{q} K(v_i) = 0 | K(v^*) = 0]) \tag{1h}$$

$$\geq \frac{1}{(n+1)m}(1 - \frac{q}{m})(1 - \sum_{i=1}^{q} \Pr[K(v_i) = 0 | K(v^*) = 0]) \tag{1i}$$

$$= \frac{1}{(n+1)m}(1 - \frac{q}{m})^2 \tag{1j}$$

$$\geq \frac{1}{(n+1)m}(1 - 2\frac{q}{m}) \tag{1k}$$

Equations 1d and 1g come from the fact that $\Pr[K(v) = 0] = \frac{1}{m}$ for any query, v. The $\frac{1}{n+1}$ factor of Equation 1e comes from the simulator taking a guess of k. Equation 1j is derived from the pairwise independence of the probabilities that $K(v) = 0$,$K(v') = 0$ for any pair of different queries v, v'. The probabilities are pairwise independent since the sums $x' + \sum_{i \in V} x_i \pmod m$ and $x' + \sum_{i \in V'} x_i \pmod m$ will differ in at least one random x_j.

We can optimize the last equation by setting $m = 4q$ (as we did in the simulation), where q is the maximum number of queries. (If the adversary makes less queries the probability of not aborting can only be greater). Solving for this gives us a lower bound $\lambda = \frac{1}{8(n+1)q}$. □

We now can calculate the distributions $\mathcal{P}_{BDH}$ and $\mathcal{R}_{BDH}$. The distribution $\mathcal{R}_{BDH}$ is simply $\frac{1}{2}$. When the simulator is given a random element as the last term in the tuple the simulator will either abort (and guess $\beta' = 1$ with prob-

ability $\frac{1}{2}$) or it will guess $\beta' = 1$ when the adversary is correct in guessing γ. However, the γ will be completely hidden from the adversary in this case so the adversary will be correct with probability $\frac{1}{2}$.

The calculation of $\mathcal{P}_{BDH}$ is somewhat more complicated. In the second simulation the adversary's view of the simulation will be identical to the real game. We want to know the probability that the guess $\beta' = 1$.

We then break the event into the abort and non-abort cases and see that $\Pr[\beta' = 1]$ is the sum of $\Pr[\beta' = 1|\texttt{abort}]\Pr[\texttt{abort}]$ and $Pr[\beta' = 1|\overline{\texttt{abort}}]\Pr[\overline{\texttt{abort}}]$. We observe that $\Pr[\beta' = 1|\texttt{abort}] = \frac{1}{2}$ and that when the simulator does not abort $\beta' = 1$ when the adversary correctly guesses $\gamma' = \gamma$. Then, we have $\mathcal{P}_{BDH} = \frac{1}{2} + \frac{1}{2}(\Pr[\overline{\texttt{abort}}|\gamma' = \gamma]\Pr[\gamma' = \gamma] - \Pr[\overline{\texttt{abort}}|\gamma' \neq \gamma]\Pr[\gamma' \neq \gamma])$. By our assumption, this is equal to $\frac{1}{2} + \frac{1}{2}(\Pr[\overline{\texttt{abort}}|\gamma' = \gamma](\frac{1}{2} + \epsilon) - \Pr[\overline{\texttt{abort}}|\gamma' \neq \gamma](\frac{1}{2} - \epsilon))$. All that is left to do is to both lower and upper bound the probability of not aborting in our simulation. We state the following claim.

Claim. If the simulator takes takes $O(\epsilon^{-2}\ln(\epsilon^{-1})\lambda^{-1}\ln(\lambda^{-1}))$ samples when computing the estimate η', then $(\frac{1}{2} + \epsilon)\Pr[\overline{\texttt{abort}}|\gamma' = \gamma] - (\frac{1}{2} - \epsilon)\Pr[\overline{\texttt{abort}}|\gamma' = \gamma] \geq \frac{3}{2}\lambda\epsilon$.

We prove the claim in Appendix A.

Plugging in the claim we have $\mathcal{P}_{BDH} \geq \frac{1}{2} + \frac{3}{4}\lambda\epsilon$. Then, $\frac{1}{2}(\mathcal{P}_{BDH} - \mathcal{R}_{BDH}) \geq \frac{3}{4}\lambda\epsilon \geq \frac{\epsilon}{32(n+1)q}$. □

We note that if there was a way for the simulator to efficiently compute the abort probability, η, for a given set of queries (as opposed to sampling) then we could improve the time component of our reduction could be significantly improved in addition to simplifying our analysis.

6 Hierarchical IBE and CCA Security

In Section 4 we discussed the similarity of our scheme to the 1-level hierarchical IBE (HIBE) scheme of Boneh and Boyen [1]. We can further take advantage of the similarity of our schemes to construct an ℓ-level HIBE scheme in an obvious manner. (For each level we must generate new parameters u' and U.)

The problem with using our techniques to construct an HIBE scheme is that the reduction becomes inefficient for all but small values of ℓ. In particular to construct a scheme in which any efficient adversary has at most ϵ advantage it must be true that all efficient adversaries have at most an $O((nq)^{\ell}\epsilon)$ advantage in the decisional BDH game. The intuition behind this is that in the simulation the setup must be "match" the challenge identity at all ℓ different levels in order to not abort. (However, our reduction still provides a stronger reduction than that of Boneh and Boyen [1] for a fully secure HIBE scheme.) For this reason we still consider the construction of a fully secure HIBE scheme without random oracles to be an open problem.

Recent results of Canetti et al. [10], further improved upon by Boneh and Katz [6], show how to build a CCA-secure Identity-Based encryption scheme

from a 2-level HIBE scheme. We can actually build a hybrid 2-level HIBE [15, 13] scheme that uses our scheme at the first level and the scheme of Boneh and Boyen [1] at the second level. Since the transformations [10, 6] only require Selective-ID security at the second level our hybrid construction is CCA secure without any significant further degradation in the security reduction relative to our non-hierarchical construction.

7 A Signature Scheme

Boneh and Franklin [5] describe a generic method for converting any Identity-Based Encryption scheme into a signature scheme. The public key of the signature scheme corresponds to the global parameters of the IBE scheme. To sign a message, M, in the signature scheme the signer gives an IBE private key of M as the signature of M. To verify a signature of M the verifier encrypts a random value, R, to the identity M, then attempts to decrypt the ciphertext with the private key. The signature is accepted if the decryption successfully decrypts to R. We note that this is a randomized verification algorithm.

In the generic transformation the security of the resulting signature scheme reduces to the security of the Identity-Based Encryption scheme. Thus, we immediately have a signature scheme which is secure as the decisional BDH problem. However, we can use the bilinear map in order to deterministically verify a signature and get a signature scheme that reduces to the weaker computational Diffie-Hellman assumption. We note that similar techniques have been used previously. For example, the signatures in the scheme of Boneh, Lynn, Shacham [7] correspond to private keys of the Boneh-Franklin IBE system. We describe our signature scheme for completeness.

7.1 Construction

Let $\mathbb{G}$ be a group of prime order, p, for which there exists an efficiently computable bilinear map into $\mathbb{G}_1$. Additionally, let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ denote the bilinear map and g be the corresponding generator. The size of the group is determined by the security parameter. We will sign messages of n bits; again, we can use a collision-resistant hash function, $H : \{0,1\}^* \rightarrow \{0,1\}^n$, to sign messages of arbitrary length.

Setup. The public key is generated as follows. A secret $\alpha \in \mathbb{Z}_p$ is chosen at random. We choose a random generator, g, and set the value $g_1 = g^\alpha$ and choose g_2 randomly in $\mathbb{G}$. Additionally, the algorithm chooses a random value $u' \in \mathbb{G}$ and a random n-length vector $U = (u_i)$, whose elements are chosen at random from $\mathbb{G}$. The published public key is g,g_1, g_2,u', and U. The secret key is g_2^α.

Signing. Let M be an n-bit message to be signed and M_i denote the ith bit of M, and $\mathcal{M} \subseteq \{1, \ldots, n\}$ be the set of all i for which $M_i = 1$. A signature of M

is generated as follows. First, a random $r \in \mathbb{Z}_p$ is chosen. Then the signature is constructed as:

$$\sigma_M = \left(g_2^{\alpha} \left(u' \prod_{i \in \mathcal{M}} u_i \right)^r, g^r \right).$$

Verification. Suppose we wish to check if $\sigma = (\sigma_1, \sigma_2)$ is a signature for a message M. The signature is accepted if $e(\sigma_1, g)/e(\sigma_2, u' \prod_{i \in \mathcal{M}} u_i) = e(g_1, g_2)$.

7.2 Security

Theorem 2. *The signature scheme is* (t, q, ϵ) *existentially unforgeable assuming the decisional-*$(t, \frac{\epsilon}{16(n+1)q})$ *BDH assumption holds, where* $\lambda = \frac{1}{8(n+1)q}$.

We omit the proof of this theorem, but note that it is analogous to the proof our IBE scheme. The fact that the adversary returns a forgery results in two important differences though. First, the forgery is used to solve the computational Diffie-Hellman problem. Secondly, since a forgery is returned there is no need for an artificial abort stage as in the previous reduction.

Other efficient schemes that are secure against existential forgery under an adaptive chosen-message attack [14] in the standard model depend upon the Strong-RSA assumption [12, 11] or the Strong Diffie-Hellman assumption [3]. Additionally Boneh, Mironov, and Shoup [8] describe a tree-based signature scheme based on the computational Diffie-Hellman assumption.

8 Conclusions

We presented the first efficient Identity-Based Encryption scheme that is secure in the full model without random oracles. We proved our the security of our scheme by reducing it to the decisional Bilinear Diffie-Hellman problem. Additionally, we showed how our Identity-Based encryption scheme can be converted to an efficient signature scheme that depends only upon the computational Diffie-Hellman assumption in the standard model.

This work motivates two interesting open problems. The first is to find an efficient Identity-Based Encryption system (without random oracles) that has short public parameters. The second, is to find an IBE system with a tight reduction in security. Such a solution would also likely permit an efficient reduction for an analogous HIBE scheme.

Acknowledgments

We would like to thank Mihir Bellare, Dan Boneh, and the anonymous reviewers of Eurocrypt 2005 for giving helpful suggestions.

References

1. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity based encryption without random oracles. In *Proceedings of the International Conference on Advances in Cryptology (EUROCRYPT '04)*, Lecture Notes in Computer Science. Springer Verlag, 2004.
2. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Proceedings of the Advances in Cryptology (CRYPTO '04)*, 2004.
3. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Berlin: Springer-Verlag, 2004.
4. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229. Springer-Verlag, 2001.
5. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.
6. Dan Boneh and Jonathan Katz. Improved efficiency for cca-secure cryptosystems built using identity based encryption. In *In Proceedings of RSA-CT 2005*, 2005.
7. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer-Verlag, 2001.
8. Dan Boneh, Ilya Mironov, and Victor Shoup. A secure signature scheme from bilinear maps. In *CT-RSA*, pages 98–110, 2003.
9. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Proceedings of Eurocrypt 2003*. Springer-Verlag, 2003.
10. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Proceedings of Eurocrypt 2004*. Springer-Verlag, 2004.
11. Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.
12. Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. *Lecture Notes in Computer Science*, 1592:123++, 1999.
13. Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pages 548–566. Springer-Verlag, 2002.
14. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
15. Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology: EUROCRYPT 2002*, pages 466–481, 2002.
16. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53. Springer-Verlag New York, Inc., 1985.

A Proof of Claim 3

In order to show that $(\frac{1}{2}+\epsilon)\Pr[\overline{\texttt{abort}}|\gamma'=\gamma] - (\frac{1}{2}-\epsilon)\Pr[\overline{\texttt{abort}}|\gamma'=\gamma] \geq \frac{3}{2}\lambda\epsilon$ we first upper bound the term $(\frac{1}{2}+\epsilon)\Pr[\overline{\texttt{abort}}|\gamma'=\gamma]$.

Let η be the probability of not aborting associated for a set of private key queries and challenge query on a particular run where $\gamma' = \gamma$. The simulator will make $O(\epsilon^{-2}\ln(\epsilon^{-1})\lambda^{-1}\ln(\lambda^{-1}))$ samples to calculate η' and we can use Chernoff bounds to show that $\Pr[\eta' > \eta(1+\frac{\epsilon}{8})] < \lambda\frac{\epsilon}{8}$. We then have

$$\Pr[\overline{\texttt{abort}}|\gamma' = \gamma] \geq (1-\lambda\frac{\epsilon}{8})\eta\frac{\lambda}{\eta(l+\frac{\epsilon}{8})} \geq \lambda(1-\frac{\epsilon}{8})^2 \geq \lambda(1-\frac{1}{4}\epsilon)$$

where the probability calculation is taken of the sampling of η. We now have

$$(\frac{1}{2}+\epsilon)\Pr[\overline{\texttt{abort}}|\gamma' = \gamma] \geq \lambda(\frac{1}{2}+\frac{3}{4}\epsilon).$$

(Note that the artificial abort stage aborts with probability $\frac{\lambda}{\max(\lambda,\eta')}$. Since $\eta(1+\frac{\epsilon}{8}) > \lambda$, we were able to ignore the maximum function.)

We now lower bound the term $(\frac{1}{2}+\epsilon)\Pr[\overline{\texttt{abort}}|\gamma' \neq \gamma]$. The simulator will make $O(\epsilon^{-2}\ln(\epsilon^{-1})\lambda^{-1}\ln(\lambda^{-1}))$ samples to calculate the estimate η' and we can use Chernoff bounds to show that $\Pr[\eta' < \eta(1-\frac{\epsilon}{8})] < \lambda\frac{\epsilon}{8}$. We then have

$$\Pr[\overline{\texttt{abort}}|\gamma' \neq \gamma] \leq \lambda\frac{\epsilon}{8} + \lambda\frac{\eta}{\eta(1-\frac{\epsilon}{8})} \leq \lambda\frac{\epsilon}{8} + \lambda(1+\frac{2\epsilon}{8}) = \lambda(1+\epsilon\frac{3}{8})$$

where the probability calculation is taken of the sampling of η. We now have

$$(\frac{1}{2}-\epsilon)\Pr[\overline{\texttt{abort}}|\gamma' = \gamma] \leq \lambda(\frac{1}{2}-\frac{3}{4}\epsilon).$$

We now see that $(\frac{1}{2}+\epsilon)\Pr[\overline{\texttt{abort}}|\gamma' = \gamma] - (\frac{1}{2}-\epsilon)\Pr[\overline{\texttt{abort}}|\gamma' = \gamma] \geq \frac{3}{2}\lambda\epsilon$. □

Tag-KEM/DEM: A New Framework for Hybrid Encryption and a New Analysis of Kurosawa-Desmedt KEM*

Masayuki Abe[1], Rosario Gennaro[2], Kaoru Kurosawa[3], and Victor Shoup[4]

[1] NTT Information Sharing Platform Laboratories, NTT Corporation, Japan
[2] IBM T.J.Watson Research Center, USA
[3] Ibaraki University, Japan
[4] New York University, USA

Abstract. This paper presents a novel framework for generic construction of hybrid encryption schemes secure against chosen ciphertext attack. Our new framework yields new and more efficient CCA-secure schemes, and provides insightful explanations about existing schemes that do not fit into the previous frameworks. This could result in finding future improvements. Moreover, it allows immediate conversion from a class of threshold public-key encryption to a hybrid one without considerable overhead, which is not possible in the previous approaches.

Finally we present an improved security proof of the Kurosawa-Desmedt scheme, which removes the original need for information-theoretic key derivation and message authentication functions. We show that the scheme can be instantiated with any computationally secure such functions, thus extending the applicability of their paradigm, and improving its efficiency.

1 Introduction

A fundamental task of cryptography is to protect the secrecy of messages transmitted over public communication lines. For this purpose we use *encryption schemes* which use some secret information (a key) to encode a message in a way that an eavesdropper cannot decode it. However, as networks become more open and accessible, it becomes apparently clear that an adversary may not be limited to eavesdropping, but may take a more active role. She may try to interact with honest parties, by, for example, sending ciphertexts to them (possibly related to the ciphertexts she intends to decrypt) and analyze their response.

* Work done while the first author is visiting IBM T.J. Watson Research Center. The fourth author was supported by NSF grant CCR-0310297. This paper is an extended abstract combining two separate results. Proofs and detailed definitions are presented in the full versions available as [3, 20].

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 128–146, 2005.

Such active attacks can be proven to be much more powerful and hard to combat than passive ones (see for example [6]).

To model this type of attacks, the notion of *chosen-ciphertext security* was introduced by Naor and Yung [22] and developed by Rackoff and Simon [24], and Dolev, Dwork, and Naor [17]. Security against a chosen ciphertext attack (CCA security, in short) means that, even if the adversary is allowed to query a *decryption oracle* on ciphertexts of her choosing, then she obtains no information about messages encrypted in other ciphertexts. The first CCA-secure cryptosystems were presented in [22, 24, 17], but they were quite impractical, as they rely on generic techniques for non-interactive zero-knowledge. In a breakthrough result, Cramer and Shoup in [12] presented the first truly practical CCA-secure cryptosystem, whose security is based on the hardness of the decisional Diffie-Hellman problem. This construction was generalized in [13], using a new cryptographic primitive called *projective hash functions*.

Public-key encryption schemes often limit the message space to a particular group, which can be restrictive when one wants to encrypt arbitrary messages. For this purpose *hybrid* schemes are devised, composed by two parts. First a *Key Encapsulation Mechanism* (KEM) is invoked: a random group element is encrypted and then mapped via a key derivation function into a random key K. Then a *Data Encapsulation Mechanism* is performed: the previous key K is used to encrypt the message using a symmetric encryption scheme. A formal treatment is found in [27, 14].

In order to obtain a CCA-secure hybrid encryption, it is sufficient that both KEM and DEM are CCA-secure. (Accordingly, we refer the framework of [27, 14] as CCA KEM/DEM framework in this paper). Recently in [21], Kurosawa and Desmedt introduced a hybrid encryption scheme which is a modification of the hybrid scheme presented in [25]. Their scheme is interesting from both a theoretical and a practical point of view. When one looks at it as a KEM/DEM scheme, we do not know if their KEM is CCA-secure, yet the resulting scheme is CCA-secure and more efficient than the one in [25] both in computation and bandwidth. Thus the Kurosawa-Desmedt scheme points out that to obtain CCA-secure hybrid encryption, requiring both KEM/DEM to be CCA-secure, while being a sufficient condition, may not be a necessary one, and might indeed be an overkill. There are other hybrid encryption schemes in the literature, e.g.,[5, 23], which are very efficient, mostly in the random oracle model, but do not fit to the CCA KEM/DEM framework.

Our Contribution. Prompted by the above observation, we set out to investigate another framework that yields more efficient hybrid encryption and captures a wider variety of existing schemes. Our results can be summarized as follows:

– We introduce Tag-KEM: a KEM which also takes as input a *tag*. Though such a notion is known in the literature, e.g., [27], we give an extended syntax and show, somewhat surprisingly, that if one uses a CCA-secure Tag-KEM in a novel way then it is sufficient for the DEM to be secure simply against passive attackers in order to yield CCA-secure hybrid encryption.

- We present several constructions of CCA-secure Tag-KEMs based on various combinations of assumptions.
- We show that the Tag-KEM/DEM framework provides a simple way to create threshold versions of CCA-secure hybrid encryption schemes, which is not possible in the CCA KEM/DEM framework.
- We show how several schemes in the literature can be casted in our Tag-KEM/DEM framework. Furthermore we show that some of those schemes can actually be simplified when considered as instances of our framework.
- Finally, we present an improved proof of the Kurosawa-Desmedt scheme. The original proof required the use of information-theoretic key derivation and message authentication functions. We show that any computationally secure such function suffices for the security of the scheme. The improvement is not just theoretical, but it has important practical implications as well. First of all it allows for a modular design in which any secure key derivation and MAC function can be used. Moreover our proof yields shorter security parameters and thus improved efficiency.

2 Definitions

2.1 Key Encapsulation Mechanism with Tags

In CCA KEM/DEM framework of [14], a KEM consists of three algorithms as public-key encryption does, except that the encryption algorithm takes only pk and outputs a random one-time key and its encryption. The encryption function may also take an extra string (called tag) as an input associated to every ciphertext. In our model, we divide the encryption function into two functions in such a way that the first one selects a random key and the second one encrypts the key along with a given tag. We call a KEM that meets this model a Tag-KEM. Formally:

$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^\lambda)$: A probabilistic algorithm that generates public-key pk and private-key sk. The public-key defines spaces for tags and encapsulated keys denoted by $\mathcal{T}$ and $\mathcal{K}_K$, respectively.

$(\omega, dk) \leftarrow \mathsf{TKEM.Key}(pk)$: A probabilistic algorithm that outputs one-time key $dk \in \mathcal{K}_D$ and internal state information ω that essentially carries dk. $\mathcal{K}_D$ is the key-space of DEM.

$\psi \leftarrow \mathsf{TKEM.Enc}(\omega, \tau)$: A probabilistic algorithm that encrypts dk (embedded in ω) into ψ along with τ, where τ is called a tag.

$dk \leftarrow \mathsf{TKEM.Dec}_{sk}(\psi, \tau)$: 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.

Note that, in the above syntactic definition, τ is not included in ψ and explicitly given to $\mathsf{TKEM.Dec}$. Such explicit treatment of τ has some notational advantages when we consider an adversary who tries to alter the tag without affecting to the ciphertext.

Tag-KEM is a generalization of KEM because if the tag is a fixed string, it is a KEM. Tags associated to PKE or KEM can be found in the literature (e.g. see [28, 27]), but their syntactic definition and the purpose are different from those of ours; A tag is supposed to carry an identity of the encryptor and has to be fixed before DEM key is selected in their definition. Despite the limitations, their particular implementation fits also to our model without essential modification.

The security of Tag-KEM requires that the adversary should fail to distinguish whether a given dk is the one embedded in cipehrtext (ψ, τ) or not, with adaptive 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 oracle machine that plays 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. $\psi \leftarrow \mathsf{TKEM.Enc}(\omega, \tau)$
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 the decryption oracle $\mathcal{O}$. Variable υ_1, υ_2 are the internal 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 define $\epsilon_{\mathrm{tkem},A_T} = \left|\Pr[\tilde{\delta} = \delta] - \frac{1}{2}\right|$ and $\epsilon_{\mathrm{tkem}} = \max_{A_T}(\epsilon_{\mathrm{tkem},A_T})$ where the maximum is taken over all machines. We say that a Tag-KEM is CCA-secure if ϵ_{tkem} is negligible in λ.

2.2 Data Encapsulation Mechanism and Public-Key Encryption

Data Encapsulation Mechanism (DEM). A DEM is a symmetric encryption scheme that consists of two algorithms, $\mathsf{DEM.Enc}$ and $\mathsf{DEM.Dec}$ such that $\mathsf{DEM.Enc}$ is an encryption algorithm that encrypts m into ciphertext χ by using symmetric-key $dk \in \mathcal{K}_D$ and $\mathsf{DEM.Dec}$ is a corresponding decryption algorithm that recovers message m from input ciphertext χ by using the same symmetric-key.

For our purpose, we only require DEM to be indistinguishable against passive attacks. Namely, adversary A_D chooses two same-length messages and given a ciphertext of either of the messages from the encryption oracle and decide which of the messages is encrypted. It is stressed that the ciphertext is made by a random key and the key is used only once. DEM is one-time secure if any polynomial-time adversary succeeds in distinguishing the encryption oracle's choice with probability at most $\frac{1}{2} + \epsilon_{\mathrm{dem}}$ where ϵ_{dem} is negligible in the security parameter. One-time pad is a simple example that fulfills this security notion.

Public-key Encryption (PKE). A public-key encryption scheme consists of key-generation algorithm $\mathsf{PKE.Gen}$, encryption algorithm $\mathsf{PKE.Enc}$, and decryption

algorithm PKE.Dec, which are defined in a standard way. We also define chosen ciphertext security for PKE in the standard sense. That is, the adversary chooses two messages from the message space, and is given a ciphertext of either of them from the encryption oracle. The adversary is also given access to the decrypiton oracle that will decrypt any ciphertext except for the one made by the encryption oracle. PKE is CCA secure if any polynomial-time adversary succeeds in distinguishing the encryption oracle's choice with probability at most $\frac{1}{2} + \epsilon_{\mathrm{pke}}$ where ϵ_{pke} is negligible in the security parameter.

3 Generic Construction of Hybrid PKE

In GAME.TKEM, it is important to see that the same ψ can be asked to the decryption oracle as long as τ is different. Therefore, to conform CCA-security, the pair (ψ, τ) must be non-malleable, which means that CCA-secure Tag-KEM provides integrity to τ. We exploit this property to protect the DEM part so as to be non-malleable.

Now in our construction of hybrid PKE, we require that Tag-KEM accepts any string as a tag, i.e., $\mathcal{T} = \{0,1\}^*$. First of all, PKE.Gen is the same as TKEM.Gen; Given security parameter λ, it outputs pubic-key pk and private-key sk. Encryption and decryption functions are as follows.

Function: $\mathsf{PKE.Enc}_{pk}(m)$

$(\omega, dk) \leftarrow \mathsf{TKEM.Key}(pk)$
$\chi \leftarrow \mathsf{DEM.Enc}_{dk}(m)$
$\psi \leftarrow \mathsf{TKEM.Enc}(\omega, \chi)$
Output $c = (\psi, \chi)$

Function: $\mathsf{PKE.Dec}(sk, c)$

$(\psi, \chi) \leftarrow c$
$dk \leftarrow \mathsf{TKEM.Dec}_{sk}(\psi, \chi)$
$m \leftarrow \mathsf{DEM.Dec}_{dk}(\chi)$
Output m

When the length of DEM key varies depending on the length of message, like one-time pad, the syntax of Tag-KEM will be modified so that TKEM.Enc and TKEM.Dec can take necessary information.

Theorem 1. *If Tag-KEM is CCA secure and DEM is one-time secure then the Hybrid PKE scheme in Section 3 is CCA secure. In particular,* $\epsilon_{\mathrm{pke}} < 2\epsilon_{\mathrm{tkem}} + \epsilon_{\mathrm{dem}}$.

Proof. Let A_E be a polynomial-time oracle machine that launches a chosen-ciphertext attack against the above hybrid encryption scheme. Let $\mathcal{O}$ denote the decryption oracle. Call this attack GAME.PKE.

[GAME.PKE]

Step 1. $(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^\lambda)$
Step 2. $(m_0, m_1, \upsilon) \leftarrow {A_E}^{\mathcal{O}}(pk)$
Step 3. $b \leftarrow \{0,1\}$, $(\omega, dk) \leftarrow \mathsf{TKEM.Key}(pk)$, $\chi \leftarrow \mathsf{DEM.Enc}_{dk}(m_b)$, $\psi \leftarrow \mathsf{TKEM.Enc}(\omega, \chi)$
Step 4. $\tilde{b} \leftarrow {A_E}^{\mathcal{O}}(\upsilon, (\psi, \chi))$

Let X denote the event that $\tilde{b} = b$ happens in GAME.PKE. The goal of this proof is to bound $\Pr[X]$. First we modify Step-3 so that DEM.Enc takes random key $dk^{\times}$ instead of the legitimate one generated by TKEM.Key. Call this game GAME.PKE'. Let X' denote the event of $\tilde{b}' = b$ in GAME.PKE'. We claim that $|\Pr[X] - \Pr[X']| \leq 2\,\epsilon_{\mathrm{tkem}}$, which is shown by constructing A_T that attacks the underlying Tag-KEM scheme by using A_E. First A_T is given public-key pk and passes it to A_E. Given m_0 and m_1 from A_E, A_T requests dk_δ to the encryption oracle of GAME.TKEM. A_T then selects $b \leftarrow \{0,1\}$ and computes $\chi = \mathsf{DEM.Enc}_{dk_\delta}(m_b)$. By sending TKEM.Enc χ as a tag, A_T receives ψ and sends ciphertext (ψ, χ) to A_E. Every decryption query from A_E is forwarded to decryption oracle TKEM.Dec. If $\perp$ is returned, it is forwarded to A_E. Otherwise, A_K decrypts χ by using the key given from oracle TKEM.Dec and pass the resulting message to A_E. When A_E outputs $\tilde{b} = b$, A_K outputs $\tilde{\delta} = 1$ meaning that dk_δ is the real key. Otherwise, if A_E outputs $\tilde{b} \neq b$, A_K outputs $\tilde{\delta} = 0$ meaning that dk_δ is random. Now observe that the view of A_E is identical to that in GAME.PKE when $\delta = 1$, and that in GAME.PKE' when $\delta = 0$. Accordingly, $\Pr[\tilde{b} = b|\delta = 1] = \Pr[X]$ and $\Pr[\tilde{b} = b|\delta = 0] = \Pr[X']$. Therefore,

$$\begin{aligned}
\Pr[\tilde{\delta} = \delta] - \frac{1}{2} &= \frac{1}{2}(\Pr[\tilde{\delta} = 1|\delta = 1] - \Pr[\tilde{\delta} = 1|\delta = 0]) \\
&= \frac{1}{2}(\Pr[\tilde{b} = b|\delta = 1] - \Pr[\tilde{b} = b|\delta = 0]) \\
&= \frac{1}{2}(\Pr[X] - \Pr[X'])
\end{aligned}$$

Since $\left|\Pr[\tilde{\delta} = \delta] - \frac{1}{2}\right| \leq \epsilon_{\mathrm{tkem}}$, we have $|\Pr[X] - \Pr[X']| \leq 2\epsilon_{\mathrm{tkem}}$.

Next, we show that A_E playing GAME.PKE' essentially conducts a passive attack to DEM, i.e., $\left|\Pr[X'] - \frac{1}{2}\right| \leq \epsilon_{\mathrm{dem}}$. It is shown by constructing A_D that plays GAME.DEM by using A_E. A_D first generates (pk, sk) by using PKE.Gen and gives pk to A_E. When m_0 and m_1 are given from A_E, A_D forwards them to encryption oracle of GAME.DEM and receives cipehrtext χ. It then computes ψ by following TKEM.Key and TKEM.Enc by using χ as a tag, and sends $c = (\psi, \chi)$ to A_E. Note that the key chosen by the encryption oracle of GAME.DEM and the one embedded in ψ are independent and randomly chosen. All decryption queries are correctly processed by using sk. When A_E outputs $\tilde{b}$, A_D outputs $\tilde{\xi} = \tilde{b}$. It is now easy to see that, in this construction, GAME.PKE' is perfectly simulated and whenever A_E wins, so does A_D. Hence $\left|\Pr[X'] - \frac{1}{2}\right| \leq \epsilon_{\mathrm{dem}}$. The major factors of the running time of A_D is that of A_E and that for simulating the decryption oracle which grows linearly in the number of decryption queries.

In summary, we have:

$$\begin{aligned}
|(\Pr[X] - \frac{1}{2}) - (\Pr[X'] - \frac{1}{2})| &\leq 2\epsilon_{\mathrm{tkem}} \\
\epsilon_{\mathrm{pke}} - \epsilon_{\mathrm{dem}} &\leq 2\epsilon_{\mathrm{tkem}} \\
\epsilon_{\mathrm{pke}} &\leq 2\epsilon_{\mathrm{tkem}} + \epsilon_{\mathrm{dem}}
\end{aligned}$$

where ϵ_{tkem} and ϵ_{dem} are assumed negligible. □

4 Construction of Tag-KEM

This section develops some methods for obtaining Tag-KEM from PKE or KEM. Note that KEM is generally obtained from PKE. Hence starting from KEM is more general.

4.1 Based on PKE with Long Plaintext

When CCA-secure PKE is available, the first idea would be to encrypt the tag as a part of the plaintext together with the DEM key to encapsulate. It indeed works well if there is enough space in a plaintext. Lengthy tags would be compressed by using a hash function. We can show that a target collision-free hash function [14], which is implied by universal one-way hash function, is sufficient for this purpose.

Formally, the construction is as follows. $\mathsf{TKEM.Gen}$ is essentially the same as $\mathsf{PKE.Gen}$; It outputs (pk, sk). It also selects hash function H. (For notational simplicity, we assume that H is included in pk and sk.) $\mathsf{TKEM.Key}$ chooses random dk from $\mathcal{K}_D$. It also outputs state information $\omega = pk||dk$. The encryption and decryption functions are as follows.

Function: $\mathsf{TKEM.Enc}(\omega, \tau)$

$(pk, dk) \leftarrow \omega$
$\tau' = H(\tau)$
$\psi = \mathsf{PKE.Enc}_{pk}(dk||\tau')$
Output ψ.

Function: $\mathsf{TKEM.Dec}_{sk}(\psi, \tau)$

$dk||\tau' \leftarrow \mathsf{PKE.Dec}(sk, \psi)$
If $\tau' = H(\tau)$, return dk.
Return $\perp$, otherwise.

The resulting Tag-KEM is as secure as attacking the underlying PKE or hash function. Let ϵ_{tch} be the success probability of finding a target collision for H. The following theorem holds.

Theorem 2. *If PKE is CCA-secure and H is target collision-free, the above Tag-KEM is CCA-secure. Especially,* $\epsilon_{\mathrm{tkem}} \leq \epsilon_{\mathrm{pke}} + \epsilon_{\mathrm{tch}}$.

The RSA-based simple KEM [27] can be seen as an instance of this method in the random oracle model. Applying Theorem 1 yields a hybrid PKE that is a special case of [15]. Also, similar hybrid PKE is found in legendary protocols such as [4].

4.2 Based on CCA-Secure KEM and MAC

In this section we present a CCA-secure Tag-KEM based on a CCA-secure KEM and a secure message authentication code (MAC). Here, MAC is assumed to be strongly unforgeable against one-time chosen message and unbound MAC verification attack. That is, a MAC adversary is given a MAC for an arbitrary message of its choice and attempts to create a valid message-MAC pair that

is different from the observed pair. The adversary also has polynomially many access to MAC verification oracle that verifies an arbitrary pair of a message and a MAC. We say MAC is one-time secure if it satisfies this security notion. Theoretically, such a MAC is available without intractability assumptions.

The idea is to encrypt a random key K using the KEM, and derive two keys dk, mk from K. The first, dk is the actual encryption key, while mk is used to MAC the tag. The resulting MAC is appended to the ciphertext. A decryptor not only checks that the KEM decryption is correct, but also checks that the MAC on the tag, using the decrypted key mk, is correct. A formal description follows.

Construction of Tag-KEM: Let $\Pi_L = (\mathsf{KEM.Gen}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$ be a KEM. Let $\mathsf{MAC} = (\mathsf{MAC.Sign}, \mathsf{MAC.Ver})$ be a MAC. Let $\mathsf{KDF}_2 : \mathcal{K}_K \to \mathcal{K}_D \times \mathcal{K}_M$ be a key derivation function where $\mathcal{K}_D$ is the key-space of DEM and $\mathcal{K}_M$ is the key-space of MAC. By using these components, we construct a Tag-KEM as follows. $\mathsf{TKEM.Gen}$ is the same as $\mathsf{PKE.Gen}$; It outputs (pk, sk). $\mathsf{TKEM.Key}$ is that, given pk, it computes $(K, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}()$ and $(dk, mk) \leftarrow \mathsf{KDF}_2(K)$. Then it outputs dk and state information $\omega = (mk, \phi)$. The encryption and decryption functions are as in the table below.

The security of KDF_2 requires that its output distribution is indistinguishable from uniform one over the key-spaces. By ϵ_{kdf}, we denote the maximum advantage over all polynomial-time distinguisher. If ϵ_{kdf} is negligible, we say that KDF_2 is secure. If KDF_2 requires a key, it is generated by $\mathsf{TKEM.Gen}$ and included in pk and sk.

Function: $\mathsf{TKEM.Enc}(\omega, \tau)$	**Function:** $\mathsf{TKEM.Dec}_{sk}(\psi, \tau)$
$(mk, \phi) \leftarrow \omega$ $\sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau)$ Output $\psi = (\phi, \sigma)$	$(\phi, \sigma) \leftarrow \psi$ $K \leftarrow \mathsf{KEM.Dec}_{sk}(\phi)$ $(dk, mk) \leftarrow \mathsf{KDF}_2(K)$ If $K = \perp$ or $\mathsf{MAC.Ver}_{mk}(\sigma, \tau) \neq 1$, output $\perp$. Otherwise, output dk.

Clearly the CCA security of the KEM scheme will prevent an adversary from gaining any advantage by manipulating the KEM ciphertext. On the other hand the security of the MAC will prevent an adversary from gaining any advantage by manipulating the MAC. The following theorem holds.

Theorem 3. *If Π_L is CCA secure, MAC is one-time secure, and KDF_2 is secure then the resulting Tag-KEM is CCA secure. In particular, $\epsilon_{\mathrm{tkem}} \leq 4\epsilon_{\mathrm{kem}} + q_D\,\epsilon_{\mathrm{mac}} + 5\epsilon_{\mathrm{kdf}}$ where q_D is the maximum number of decryption queries.*

Applying Theorem 1 to the above Tag-KEM yields the same hybrid encryption scheme as in CCA KEM/DEM framework. But by analysing the same scheme in our framework, we can show that CCA KEM is an overkill. In [3], it is shown that there exists a class of KEM that is strictly weaker than CCA but suffices for this construction.

4.3 Based on KEM with Hash Function

We show another approach that might be available when the underlying PKE does not have enough plaintext length as needed in Section 4.1 and/or increasing ciphertext length as in Section 4.2 is not acceptable.

If a KEM uses a hash function, probably for integrity of ciphertext or plaintext, the KEM may be converted to a Tag-KEM simply by including the tag into the hash function. This approach is correct if the hash function is involved in the scheme in a 'meaningful' way and provides 'sufficient' security. Although generic construction that follows formal version of these intuitive terms can be shown, it does not seem quite useful due to its complexity. Showing that a KEM fits to the generic framework may not be simpler than directly proving that the resulting Tag-KEM scheme is secure. Indeed, in all cases we have in mind, the security proof can be done by minor or obvious modification of that of the original KEM (or PKE). Therefore, we only show two concrete constructions of Tag-KEM based on well known encryption schemes; OAEP+ [26] and Cramer-Shoup encryption [12]. In the following, the description of the original schemes are obtained just by dropping the tag τ.

From OAEP+. Let f be a one-way trapdoor permutation. OAEP+ encrypts dk with tag τ into ciphertext ψ in the following way:

$$r' = H'(r||dk||\tau),\ s = (G(r) \oplus dk)||r',\ w = H(s) \oplus r,\ \psi = f(s||w)$$

where r and r' are random and G, H, H' are random oracles [5].

Security is argued in the same way as the original one except the case that, for challenge ciphertext (ψ, τ) the adversary finds another valid ciphertext (ψ, τ'). Since ψ uniquely identifies r, r' and K, (ψ, τ') is valid only if $H'(r||dk||\tau) = H'(r||dk||\tau')$ holds. When H' outputs a k_1-bit string, such an event happens with probability at most $q_{H'}\, 2^{-k_1}$ where $q_{H'}$ is the maximum number of queries to H'. Based on this observation, we define game GAME.0' where decryption oracle returns $\perp$ for all queries that differs only in the tag part with the challenge ciphertext. The rest of the security proof is done in the same way as in the original paper [26] except for obvious modifications. Accordingly, only $q_{H'}\, 2^{-k_1}$ is an extra reduction cost to that of OAEP+.

From Cramer-Shoup Encryption. A Tag-KEM scheme based on Cramer-Shoup encryption over a multiplicative group, say G_q, of prime order q is the following. A private-key is $(x_1, x_2, y_1, y_2, z_1, z_2) \in Z_q$ and the public-key is $g_1, g_2 \leftarrow G_q^2$, and $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, $h = g_1^{z_1} g_2^{z_2}$. The encryption function yields $dk = h^r$ where r is random, and ciphertext (u_1, u_2, v) such that

$$u_1 = g_1^r,\ u_2 = g_2^r,\ \alpha = H(u_1||u_2||\tau),\ v = c^r d^{\alpha r}$$

where H is a hash function. Decryption first checks if $v \stackrel{?}{=} u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$ and then recovers $dk = u_1^{z_1} u_2^{z_2}$. Applying Theorem 1 results in the hybrid PKE briefly mentioned in [12].

In contrast to [12] where H can be Target Collision Free, we need slightly stronger assumption to prove the security in our framework, which nevertheless has little practical impact. We say that H is *Random Prefix Collision-Free* if any adversary wins the following game with at most negligible probability. The adversary is first given H and outputs τ and then given random x and finally outputs x' and τ' such that $H(x||\tau) = H(x'||\tau')$. We can prove that the above scheme is secure Tag-KEM when H is random prefix collision free.

It holds that (Collision-Free) $\Rightarrow$ (Random Prefix Collision-Free) $\Rightarrow$ (Target Collision-Free). Hence it is reasonable to use cryptographic hash functions like SHA-1 which can be assumed collision-free. Nevertheless, we stress that random prefix collision-freeness may not necessarily be equivalent to collision-free because, for example, it is not clear how to perform a birthday attack in the above game (if the randomness of x affects to the output). Theoretically, we do not know constructions of random prefix collision-free hash functions from target collision-free or universal one-way hash functions, thus we resort to strong collision-freeness. The only drawback is that this requires a longer output (about twice as much because the birthday paradox applies here), but that does not affect our construction.

4.4 Based on ID-Based PKE

An ID-based encryption scheme is selective-ID secure when it is secure against chosen ciphertext and chosen ID attacks provided that the target ID is committed at the beginning and the ID must not be included in any decryption query. It is shown in [10] that selective-ID ID-based encryption schemes (sIBE in short) can be strengthened to a full CCA secure ones by using one-time signature. Then, according to CCA KEM/DEM framework, an ID-based hybrid encryption scheme can be obtained by combining it with a CCA secure DEM. We show that the conversion from sIBE to full IBE also yields a Tag-KEM. Accordingly, the DEM part can be simplified to be a one-time secure DEM. The resulting scheme yields shorter ciphertexts than before.

Let $(\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$ be a one-time signature scheme where $\mathsf{SIG.Gen}$ is a key generation algorithm, $\mathsf{SIG.Sign}$ is a signature generation algorithm, and $\mathsf{SIG.Ver}$ is a signature verification algorithm. Let $\mathsf{sIBE.Enc}(pk, \mathrm{ID}, m)$ be the encryption function of an sIBE. Then, we construct a Tag-KEM scheme as follows: It encrypts (pk, dk) and τ into ciphertext $\psi = (vk, \phi, \sigma)$ where

$$(vk, sk) \leftarrow \mathsf{SIG.Gen}(1^\lambda),\ \phi \leftarrow \mathsf{sIBE.Enc}(pk, vk, dk),\ \sigma = \mathsf{SIG.Sign}(sk, \phi||\tau).$$

Including τ into the message to be signed provides integrity to the tag without affecting the security of the original scheme. Indeed, the security proof is almost the same as in [10] with obvious modification. The reduction cost does not change, either. One can extend the above Tag-KEM to ID-based Tag-KEM in the same way starting from a 2nd-level ID Encryption function that takes two ID's. (A given ID is assigned to the first ID and vk is assigned to the second ID.) For efficient implementations of sIBE based on standard cryptographic assumptions, we refer to [7].

In [8], Boneh and Katz improved the efficiency of [10] by replacing the one-time signature with commitment scheme (using hash function) and MAC. Part of their scheme can also be seen as a Tag-KEM.

5 Applications

5.1 Threshold Hybrid PKE

Designing a threshold hybrid PKE is not a trivial task. Even though threshold PKE is available, it is not clear how it can be extended to hybrid threshold PKE. By following CCA KEM/DEM framework, one will suffer from sharing KDF and $\mathsf{MAC.Ver}$, which are often implemented by number-theoretically unstructured primitives. Although these tasks are feasible using generic techniques from multi-party computation, we are focusing on efficient and practical solutions.

Since Tag-KEM/DEM framework allows the DEM part to be CPA, it immediately yields a threshold hybrid PKE once a shared Tag-KEM is available. Decrypting the DEM part is a local task. By defining CCA security for threshold PKE and DEM as in [28, 18], we can translate and prove Theorem 1 in the threshold setting. Accordingly, one can concentrate on constructing threshold Tag-KEM. A threshold KEM or PKE can be converted into a threshold Tag-KEM by following the construction in Section 4.3 or 4.1 without considerable overheads.

Threshold Cramer-Shoup encryption, secure against static adversaries, is shown in [1, 9], and the conversion technique in Section 4.3 (or result of section 4.1 with larger security parameter) can be used to obtain a threshold Cramer-Shoup Tag-KEM. Accordingly, by following the threshold version of Theorem 1, one can have a secure threshold hybrid encryption scheme in the standard model. Adaptive security can be achieved as well based on the adaptively secure threshold Cramer-Shoup encryption of [2].

5.2 Refined Fujisaki-Okamoto Conversion and More

We revisit the Fujisaki-Okamoto conversion [19] that provides secure construction of hybrid encryption in the random oracle model. By fitting their scheme into Tag-KEM/DEM framework, we can see that one of their assumptions can be eliminated and a refined version is obtained without loss of efficiency.

Let $\mathsf{PKE.Enc}_{pk}(\cdot\,;\cdot)$ be public-key encryption function where the last argument denotes a random coin used in the function. Fujisaki-Okamoto conversion combines PKE and DEM by using two random oracles, H and G, as follows:

$$\psi \leftarrow \mathsf{PKE.Enc}_{pk}(K; H(K||m)), \chi \leftarrow \mathsf{DEM.Enc}_{G(K)}(m).$$

A ciphertext is (ψ, χ). The resulting hybrid PKE is CCA-secure if PKE is one-way and DEM is one-time secure and $\mathsf{DEM.Enc}$ is a bijection between ciphertexts and messages for every fixed key.

Now one can observe that $\mathsf{PKE.Enc}_{pk}(K; H(K||\tau))$ works as a Tag-KEM encryption function that encapsulates DEM key $G(K)$. Then, according to Tag-KEM/DEM framework, we have slightly modified hybrid encryption:

$$\psi \leftarrow \mathsf{PKE.Enc}_{pk}(K; H(K||\chi)), \chi \leftarrow \mathsf{DEM.Enc}_{G(K)}(m)$$

which does not require DEM.Enc to be a bijection.

Similar observation applies to Bellare-Rogaway scheme [5], which is a special case of Fujisaki-Okamoto construction, and REACT-RSA [23].

5.3 Revisiting RCCA-Secure PKE

This section revisits RCCA-secure PKE in [11] and show that their construction of CCA-secure hybrid PKE from RCCA-secure PKE can be improved by following our Tag-KEM/DEM framework.

The notion of RCCA-secure PKE is introduced in [11]. RCCA is a variant of CCA where the decryption oracle returns a special nonce 'test' when it receives a ciphertext that yields one of the questioned message, m_0 and m_1. Accordingly, even if the adversary can tweak the challenge ciphertext without affecting the embedded plaintext (such a feature is called benign-malleability [27]), sending it to the decryption oracle will give no advantage to the adversary in determining which of the questioned messages is hidden there. 'R' stands for 'replayable' in this sense. RCCA-security is a strict relaxation of CCA-security and proven useful for several cryptographic tasks, though, currently, there is no known instance of RCCA-secure PKE that is more efficient than known CCA-secure ones.

In [11], it is shown that combining RCCA-secure PKE and CCA-secure symmetric encryption can yield CCA-secure hybrid PKE. Suppose that a CCA-secure symmetric encryption is made by combining passively secure DEM and one-time MAC. Then, their construction is summarized as follows. Given message m, output ciphertext (ϕ, χ, σ) such that;

$$\phi \leftarrow \mathsf{PKE.Enc}_{pk}(dk||mk),\ \chi \leftarrow \mathsf{DEM.Enc}_{dk}(m||\phi),\ \sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\chi)$$

where dk and mk, are chosen randomly from appropriate domains. It is stressed that ϕ is encrypted by DEM and this double-encryption structure is essential in their security proof. Due to this special structure, the construction does not fit to Tag-KEM/DEM framework. Below, we show a slightly more efficient variant that avoids double encryption and fits to Tag-KEM/DEM framework.

$$\phi \leftarrow \mathsf{PKE.Enc}_{pk}(dk||mk),\ \chi \leftarrow \mathsf{DEM.Enc}_{dk}(m),\ \sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\chi||\phi)$$

Intuitively, applying MAC to ϕ offsets the benign-malleability of ϕ. The modified scheme yields shorter ciphertexts.

From the above, we derive a Tag-KEM scheme which is summarized as follows.

$$(K, \phi) \leftarrow \mathsf{KEM.Enc}_{pk}(),\ (dk, mk) \leftarrow KDF_2(K),\ \sigma \leftarrow \mathsf{MAC.Sign}_{mk}(\tau||\phi)$$

It can be seen as a variant of the construction shown in Section 4.2; MAC is applied to $\tau||\phi$ rather than to τ.

By defining RCCA-security for KEM in the same way as that for PKE, the following theorem can be proven.

Theorem 4. *If KEM is RCCA-secure, MAC is one-time secure, and DEM is secure, the above Tag-KEM is CCA-secure. Especially,* $\epsilon_{\mathrm{tkem}} \leq 2\epsilon_{\mathrm{rkem}} + (q_D + 3)\epsilon_{\mathrm{kdf}} + \frac{q_D}{2}\epsilon_{\mathrm{mac}}$

According to Theorem 1, the modified hybrid PKE is CCA-secure. This uncovers the superfluousness of the double-encryption in the original construction and obtains a more efficient scheme.

6 New Proof for Kurosawa-Desmedt Scheme

Let us briefly recall the Kurosawa-Desmedt scheme from [21]. The group G, the hash function H and the public and secret key are as in the Cramer-Shoup scheme described earlier. It also uses a key derivation function KDF, such that for $v \in G$, $\mathsf{KDF}(v) = (k, K)$, where k is a message authentication key, and K is a symmetric encryption key.

Encryption of $m \in \{0,1\}^*$**:**

$$r \leftarrow \mathbb{Z}_q,\ u_1 \leftarrow g_1^r \in G,\ u_2 \leftarrow g_2^r \in G,\ \alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q$$
$$v \leftarrow c^r d^{r\alpha} \in G,\ (k, K) \leftarrow \mathsf{KDF}(v),\ e \leftarrow E_K(m),\ t \leftarrow \mathsf{MAC}_k(e)$$
$$\text{output } C := (u_1, u_2, e, t)$$

Decryption of $C = (u_1, u_2, e, t)$**:**

$$\alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q,\ v \leftarrow u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \in G,\ (k, K) \leftarrow \mathsf{KDF}(v)$$
$$\text{if } t \neq \mathsf{MAC}_k(e) \text{ then } \textit{reject}$$
$$\text{else output } m \leftarrow D_K(e)$$

It is possible to formalize this scheme as a Tag-KEM protocol. Indeed we can consider (u_1, u_2, t) as the Tag-KEM part (where (u_1, u_2) is the proper KEM part, e is the tag and t is a MAC on it), while e is the one-time DEM. This analysis seems identical to the one in Section 4.2, but here the basic KEM is not known to be CCA secure, so we can't invoke Theorem 3, and a proof specifically for this case is required.

The proof of security in [21] requires the MAC and KDF functions to be *information-theoretically secure*, i.e. if $v \in G$ is random, then at least the first component k of the output of $\mathsf{KDF}(v)$ should be (statistically close to) uniform; also for all e and t, if k is chosen at random, then $\Pr[\mathsf{MAC}_k(e) = t]$ is negligible. Our new proof of security, relaxes the above assumptions as follows: (i) if $v \in G$ is random, then at least the first component k of the output of $\mathsf{KDF}(v)$ should be computationally indistinguishable from uniform; (ii) the MAC function should be unforgeable. As we pointed out in the introduction this has a significant practical impact on the scheme.

Our proof shows that the Tag-KEM described above is CCA-secure. Using Theorem 1 we get that the hybrid scheme is CCA-secure as well.

Game 0. We start the proof by defining a game, called *Game 0*, which is an interactive computation between an *adversary* and a *simulator*. This game is simply the usual game used to define CCA security for Tag-KEM, in which the simulator provides the adversary's environment.

Initially, the simulator runs the key generation algorithm, obtaining the description of G, generators g_1 and g_2, keys for KDF and H (if any), along with the values $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$ and $c, d \in G$. The simulator gives the public key to the adversary.

During the execution of the game, the adversary makes a number of "decryption requests." Assume these requests are $C^{(1)}, \ldots, C^{(Q)}$, where $C^{(i)} = (u_1^{(i)}, u_2^{(i)}, e^{(i)}, t^{(i)})$. For each such request, the simulator decrypts the given ciphertext, and gives the adversary the result. We denote by $\alpha^{(i)}$, $v^{(i)}$, $k^{(i)}$, and $K^{(i)}$ the corresponding intermediate quantities computed by the decryption algorithm on input $C^{(i)}$. The oracle returns $K^{(i)}$ to the adversary.

The adversary may also make a single "challenge request." When such request is issued, the Tag-KEM encryption oracle generates $u_1 = g_1^r, u_2 = g_2^r, \alpha = H(u_1, u_2), v = c^r d^{r\alpha}$ and sets $(k_1, K_1) = \mathsf{KDF}(v)$. It also generates K_0 at random, and a random bit δ. The value K_δ is returned to the adversary who then produces a tag e and receives back (u_1, u_2, t) where $t = \mathsf{MAC}_{k_1}(e)$.

The only restriction on the adversary's requests is that after it makes a challenge request, subsequent decryption requests must be different from (u_1, u_2, e, t). At the end of the game, the adversary outputs $\hat{\delta} \in \{0, 1\}$.

Let X_0 be the event that $\hat{\delta} = \delta$. Security means that $|\Pr[X_0] - 1/2|$ should be negligible.

We prove this by considering other games, *Game 1*, *Game 2*, etc. These games will be quite similar to Game 0 in their overall structure, and will only differ from Game 0 in terms of how the simulator works. However, in each game, there will be well defined bits $\hat{\delta}$ and δ, so that in Game i, we always define X_i to the event that $\hat{\delta} = \delta$ in that game. All of these games should be viewed as operating on the same underlying probability space.

Before moving on, we make a couple of additional assumptions about the internal structure of Game 0 that will be convenient down the road. First, the simulator computes v as $(u_1)^{x_1 + y_1\alpha}(u_2)^{x_2 + y_2\alpha}$. This change is purely conceptual, since v has the same value either way. Second, we assume that g_2 is computed as $g_2 := g_1^w$ for $w \in_R \mathbb{Z}_q^*$. Second, we assume that the quantities r, u_1, u_2, α, v, k, and K_0, K_1 are computed at the very start of the game (they do not depend on values provided later by the adversary, so this can be done).

Game 1. This is the same as Game 0, except for the following differences. If the adversary ever submits $C^{(i)}$ for decryption with $(u_1^{(i)}, u_2^{(i)}) \neq (u_1, u_2)$ and $\alpha^{(i)} = \alpha$, the simulator *rejects* the given ciphertext.

In Game 1, the simulator may reject ciphertexts that would not have been rejected in Game 0. Let us call **Rejection Rule 0** the rule by which ciphertexts are rejected as in the ordinary decryption algorithm (i.e., the message authentication tags do not match). Let us call **Rejection Rule 1** this new rejection rule, introduced in Game 1.

Let F_1 be the event that the simulator applies Rejection Rule 1 in Game 1 to a ciphertext to which Rejection Rule 0 does not apply. Because Game 0 and Game 1 proceed identically until the this event occurs, we have

$$|\Pr[X_0] - \Pr[X_1]| \leq \Pr[F_1] \quad \text{and} \quad \Pr[F_1] \leq \epsilon_{\mathrm{tcr}}, \tag{1}$$

where ϵ_{tcr} is the success probability that one can find a collision in H using resources similar to those of the given adversary. By assumption, ϵ_{tcr} is negligible.

Game 2. Now generate u_2 as $g_2^{r'}$ where $r' \in_R \mathbb{Z}_q$. We have

$$|\Pr[X_2] - \Pr[X_3]| \leq \epsilon_{\mathrm{ddh}}, \tag{2}$$

where ϵ_{ddh} is the advantage with which one can solve the DDH problem, using resources similar to those of the given adversary. By assumption, ϵ_{ddh} is negligible.

Game 3. In this game, the simulator makes use of the value $w \in \mathbb{Z}_q$, where $g_2 = g_1^w$. The simulator did not need to make explicit use of this value in previous games. Indeed, we could not have used the DDH assumption if the simulator had to use w. However, we are now finished with the DDH assumption, and so the simulator is free to make use of w in this and subsequent games.

Game 3 is the same as Game 2, except that we introduce a new **Rejection Rule 2**: in responding to decryption requests, the simulator *rejects* any ciphertext $C^{(i)}$ such that $(u_1^{(i)})^w \neq u_2^{(i)}$, which is equivalent to saying that $\log_{g_1} u_1^{(i)} \neq \log_{g_2} u_2^{(i)}$.

Define F_4 to be the event that a ciphertext is rejected during Game 3 using Rejection Rule 2 to which Rejection Rules 0 and 1 are not applicable.

Clearly, we have

$$|\Pr[X_3] - \Pr[X_4]| \leq \Pr[F_4], \tag{3}$$

and we want to show that $\Pr[F_4]$ is negligible.

We postpone this until later. This is the step that allows us to avoid a circular argument in the original Kurosawa-Desmedt proof and forced them to make the information theoretic assumptions. Instead of attempting to bound $\Pr[F_4]$ right now, we shall patiently wait until Game 5, where it will be much easier. However, at this point we augment Game 3 just slightly: the simulator chooses $j \in \{1, \ldots, Q\}$, and we define F_4' to be the event that in Game 3, Rejection Rules 0 and 1 do not apply to $C^{(j)}$, but Rejection Rule 2 does apply to $C^{(j)}$. Clearly,

$$\Pr[F_4] \leq Q \Pr[F_4'], \tag{4}$$

and so it suffices to show that $\Pr[F_4']$ is negligible.

Game 4. Moving from Game 3 to Game 4 is a bit involved technically, yet the basic idea is *exactly* the same as that underlying the analysis in [12] of the original Cramer-Shoup encryption scheme. To motivate Game 4, we begin with

some observations about Game 3. Let $x := x_1 + wx_2$ and $y := y_1 + wy_2$. Then we have $c = g_1^x$ and $d = g_1^y$. Also, for $i = 1, \ldots, Q$, if $\log_{g_1} u_1^{(i)} = \log_{g_2} u_2^{(i)}$ $v^{(i)} = u_1^{x+y\alpha^{(i)}}$. Moreover, v is uniformly distributed over G, independently of x and y. Further, if $\alpha^{(j)} \neq \alpha$ and $\log_{g_1} u_1^{(j)} \neq \log_{g_2} u_2^{(j)}$ then $v^{(j)}$ is uniformly distributed over G, independently of x, y, and v. These observations follow from simple linear algebra considerations, as in [12].

Based on these observations, in Game 4, we compute a number of quantities in a different, but equivalent, manner. Let $\bar{x}, \bar{y}$ be random elements of $\mathbb{Z}_q$, and let $\bar{v}_1, \bar{v}_2$ be random elements of G. Let $(\bar{k}_i, \bar{K}_i) := \mathsf{KDF}(\bar{v}_i)$.

The key generation algorithm is modified as follows: $c \leftarrow g_1^{\bar{x}}, d \leftarrow g_1^{\bar{y}}$. The values k_1 and K_1 are set equal to $(\bar{k}_1, \bar{K}_1)$.

In processing decryption requests, for a given $C^{(i)}$ that is not subject to Rejections Rules 1 or 2, the value $v^{(i)}$ is computed as $(u_1^{(i)})^{\bar{x}+\bar{y}\alpha^{(i)}}$. Finally, we define the event F_5' to be the event in Game 4 that $C^{(j)}$ is subject to Rejection Rule 2, $C^{(j)}$ is not subject to Rejection Rule 1, and

- $(u_1^{(j)}, u_2^{(j)}) = (u_1^*, u_2^*)$ and $t^{(j)} = \mathsf{MAC}_{\bar{k}_1}(e^{(j)})$, or
- $(u_1^{(j)}, u_2^{(j)}) \neq (u_1^*, u_2^*)$ and $t^{(j)} = \mathsf{MAC}_{\bar{k}_2}(e^{(j)})$.

Note that the values $x_1, x_2, y_1, y_2, v^*, v^{(j)}$ are not used in Game 4.

We claim that

$$\Pr[X_4] = \Pr[X_5] \text{ and } \Pr[F_4'] = \Pr[F_5']. \tag{5}$$

This follows from the observations above — we have simply replaced one set of random variables by another set with same joint distribution.

It is perhaps helpful at this point to state how Game 4 works, starting from scratch:

- The simulator generates the description of G, along with a random generator g_1, and any keys for KDF and H. It computes $w, r, r', \bar{x}, \bar{y} \in_R \mathbb{Z}_q^*$, $g_2 := g_1^w$, $c := g_1^{\bar{x}}$, $d := g_1^{\bar{y}}$, $u_1 := g_1^r$, $u_2 := g_1^{wr'}$, $\bar{v}_1, \bar{v}_2 \in_R G$, $(\bar{k}_i, \bar{K}_i) \leftarrow \mathsf{KDF}(\bar{v}_i)$ and $j \in_R [1..Q]$.
  The simulator gives the description of G, the generators g_1 and g_2, keys for KDF and H (if any), along with c and d to the adversary.
- In processing a decryption request $C^{(i)} = (u_1^{(i)}, u_2^{(i)}, e^{(i)}, t^{(i)})$, the simulator first checks if $(u_1^{(i)})^w \neq u_2^{(i)}$; if so, the ciphertext is rejected. Otherwise, the simulator computes $\alpha^{(i)} := H(u_1^{(i)}, u_2^{(i)})$ and checks if $(u_1^{(i)}, u_2^{(i)}) \neq (u_1, u_2)$ and $\alpha^{(i)} = \alpha$; if so, the ciphertext is rejected. Otherwise, the simulator computes $v^{(i)}$ as $u_1^{\bar{x}+\bar{y}\alpha^{(i)}}$ and $(k^{(i)}, K^{(i)}) \leftarrow \mathsf{KDF}(v^{(i)})$. It then tests if $t^{(i)} = \mathrm{mac}_{k^{(i)}}(e^{(i)})$; if not, the ciphertext is rejected. Otherwise, the simulator returns $D_{K^{(i)}}(e^{(i)})$ to the adversary.
- In processing the challenge request, the simulator sets $K_1 = \bar{K}_1$, then chooses a random key K_0 and a random bit δ and gives K_δ to the adversary who responds with a tag e. Now the simulator computes $t \leftarrow \mathsf{MAC}_{\bar{k}_1}(e)$, and gives $C := (u_1, u_2, t)$ to the adversary.

Note that the values j and $\bar{v}_2$ (and the derived values $\bar{k}_2$ and $\bar{K}_2$) are not used in this game, other than to define the event F'_5.

Game 5. This is the same as Game 4, except that instead of applying KDF to derive the keys $\bar{k}_1, \bar{K}_1, \bar{k}_2, \bar{K}_2$, these keys are simply generated at random. Define the event F'_6 in Game 5 in the same way as it was defined in Game 4.

It is easy to see that

$$|\Pr[X_5] - \Pr[X_6]| \leq 2\epsilon_{\mathrm{kdf}} \text{ and } |\Pr[F'_5] - \Pr[F'_6]| \leq 2\epsilon_{\mathrm{kdf}}, \tag{6}$$

where ϵ_{kdf} is the advantage of distinguishing the output of the KDF from a random key pair, using resources similar to those of the given adversary. The factor of 2 comes from applying a standard "hybrid" argument to the two KDF outputs to be distinguished in moving from Game 4 to Game 5. By assumption, ϵ_{kdf} is negligible.

We claim that

$$\Pr[X_6] = 1/2 \tag{7}$$

This follows by construction — note that the key $\bar{K}_1$ in Game 5 is random, and is not used at all in the game, other than to define K_1. Therefore, conditioned on either $\delta = 0$ or $\delta = 1$, the adversary's view has the same conditional distribution; from this, it follows that the distribution of δ is independent of the adversary's view.

We also claim that

$$\Pr[F'_6] \leq 2\epsilon_{\mathrm{mac}}, \tag{8}$$

where ϵ_{mac} is the probability of breaking the message authentication code, using resources similar to those of the given adversary. This also follows by construction — one has to make a simple "hybrid" argument to account for the fact that we are breaking one out of two message authentication schemes (one keyed with $\bar{k}_1$ and the other keyed with $\bar{k}_2$, whence the factor of 2). By assumption, ϵ_{mac} is negligible.

We are now in a position to complete the proof of security. By using Eqs. (4), (5), (6), (8), we get

$$\Pr[F_4] \leq Q(2\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}). \tag{9}$$

Finally, combining (1), (2), (3), (5), (6), (7), and (9), we have:

$$|\Pr[X_0] - 1/2| \leq \epsilon_{\mathrm{tcr}} + \epsilon_{\mathrm{ddh}} + 2\epsilon_{\mathrm{kdf}} + Q(2\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}). \tag{10}$$

By assumption, the right-hand side of (10) is negligible, which finishes the proof.

Acknowledgments

The authors would like to thank Hugo Krawczyk, Shai Halevi, Mario Di Raimondo, Yevgeniy Dodis and Eiichiro Fujisaki for valuable discussion.

References

1. M. Abe. Robust distributed multiplication without interaction. In *CRYPTO '99*, LNCS 1666, pages 130–147. Springer-Verlag, 1999.
2. M. Abe and S. Fehr. Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. IACR ePrint Archive 2004/119, 2004. Preliminary version apepars in *CRYPTO '04*, LNCS 3152, pages 317–334. Springer-Verlag, 2004.
3. M. Abe, R. Gennaro and K. Kurosawa. Tag-KEM/DEM: A new framework for hybrid encryption. IACR ePrint Archive 2005/027, 2005.
4. M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudic, E. Van Herreweghen and M. Waidner. Design, implementation and Deployment of the *i*KP secure electronic payment system. *IEEE JSAC*, vol. 18, No. 4, April 2000.
5. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st ACM CCCS*, pages 62–73. Association for Computing Machinery, 1993.
6. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *CRYPTO '98*, LNCS 1462, pages 1–12. Springer-Verlag, 1998.
7. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption. In *EUROCRYPT '04*, LNCS 3027, pages 223–238. Springer-Verlag, 2004.
8. D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. IACR ePrint archive, 2004/261, 2004.
9. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT '99*, LNCS 1592, pages 90–106. Springer-Verlag, 1999.
10. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT 2004*, LNCS 3027, pages 207–222. Springer-Verlag, 2004.
11. R. Canetti, H. Krawczyk, and J. Nielsen. Relaxing chosen-ciphertext security. IACR ePrint archive, 2003/174, 2003.
12. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98*, LNCS 1462, pages 13–25. Springer-Verlag, 1998.
13. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT '02*, LNCS 2332, pages 45–64. Springer-Verlag, 2002.
14. 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.
15. Y. Dodis and J. An. Concealment and Its Applications to Authenticated Encryption. In *EUROCRYPT '03*, LNCS 2656, pages 312–329, Springer-Verlag, 2003.
16. Y. Dodis, R. Gennaro, J. Haastad, H. Krawczyk, and T. Rabin. Randomness extraction and key derivation using the CBC, Cascade and HMAC modes. In *CRYPTO '04*, LNCS 3152, pages 494–510. Springer-Verlag, 2004.
17. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In 23*rd STOC*, pages 542–552, New York City, 1991.
18. P. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *Asiacrypt 2001*, LNCS 2248, pages 351–368. Springer-Verlag, 2001.

19. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO '99*, LNCS 1666, pages 537–554. Springer-Verlag, 1999.
20. R. Gennaro and V. Shoup. A note on an encryption scheme of Kurosawa and Desmedt. IACR ePrint archive, 2004/194, 2004.
21. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO 2004*, LNCS 3152, pages 426–442. Springer-Verlag, 2004.
22. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In 22*nd STOC*, pages 427–437, 1990.
23. T. Okamoto and D. Pointcheval. REACT: Rapid enhanced-security asymmetric cryptosystem transform. In *RSA '2001*, LNCS, Springer-Verlag, 2001.
24. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO '91*, LNCS 576, pages 433–444. Springer-Verlag, 1992.
25. V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *EUROCRYPT 2000*, LNCS 1807, pages 275–288. Springer-Verlag, 2000.
26. V. Shoup. OAEP reconsidered. In *CRYPTO 2001*, LNCS 2139, pages 239–259. Springer-Verlag, 2001.
27. V. Shoup. ISO 18033-2: An emerging standard for public-key encryption (committee draft). Available at `http://shoup.net/iso/`, June 3 2004.
28. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *EUROCRYPT '98*, LNCS 1403, pages 1–16. Springer-Verlag, 1998.

Secure Remote Authentication Using Biometric Data

Xavier Boyen[1], Yevgeniy Dodis[2,⋆], Jonathan Katz[3,⋆⋆], Rafail Ostrovsky[4,⋆⋆⋆], and Adam Smith[5]

[1] Voltage, Inc.
xb@boyen.org
[2] New York University
dodis@cs.nyu.edu
[3] University of Maryland
jkatz@cs.umd.edu
[4] UCLA
rafail@cs.ucla.edu
[5] Weizmann Institute
asmith@csail.mit.edu

Abstract. Biometric data offer a potential source of high-entropy, secret information that can be used in cryptographic protocols provided two issues are addressed: (1) biometric data are not uniformly distributed; and (2) they are not exactly reproducible. Recent work, most notably that of Dodis, Reyzin, and Smith, has shown how these obstacles may be overcome by allowing some auxiliary public information to be reliably sent from a server to the human user. Subsequent work of Boyen has shown how to extend these techniques, in the random oracle model, to enable unidirectional authentication from the user to the server without the assumption of a reliable communication channel.

We show two efficient techniques enabling the use of biometric data to achieve *mutual* authentication or authenticated key exchange over a completely insecure (i.e., adversarially controlled) channel. In addition to achieving stronger security guarantees than the work of Boyen, we improve upon his solution in a number of other respects: we tolerate a broader class of errors and, in one case, improve upon the parameters of his solution and give a proof of security in the standard model.

1 Using Biometric Data for Secure Authentication

Biometric data, as a potential source of high-entropy, secret information, have been suggested as a way to enable strong, cryptographically-secure authentica-

⋆ Supported by NSF CAREER award 0133806 and Trusted Computing grant 0311095.

⋆⋆ Supported by NSF CAREER award 0447075 and Trusted Computing grants 0310751 and 0310499.

⋆⋆⋆ Supported in part by a gift from Teradata, an Intel equipment grant, an OKAWA research award, and an NSF Cybertrust grant.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 147–163, 2005.

tion of human users without requiring them to remember or store traditional cryptographic keys. Before such data can be used in existing cryptographic protocols, however, two issues must be addressed: first, biometric data are *not uniformly distributed* and hence do not offer provable security guarantees if used as is, say, as a key for a pseudorandom function. While the problem of non-uniformity can be addressed using a hash function, viewed either as a random oracle [2] or a strong extractor [20], a second and more difficult problem is that biometric data are *not exactly reproducible*, as two biometric scans of the same feature are rarely identical. Thus, traditional protocols will not even guarantee correctness when the parties use a shared secret derived from biometric data.

Much work has focused on addressing these problems in efforts to develop secure techniques for biometric authentication [8, 15, 19, 14, 22, 21]. Most recently, Dodis, Reyzin, and Smith [9] showed how to use biometric data to securely derive cryptographic keys which could then be used, in particular, for the purposes of authentication. Roughly speaking (see Section 2 for formal definitions), they introduce two primitives: a *secure sketch* which allows recovery of a shared secret given a close approximation thereof, and a *fuzzy extractor* which extracts a uniformly distributed string s from this shared secret in an error-tolerant manner. Both primitives work by constructing a "public" string pub which is stored by the server and transmitted to the user; loosely speaking, pub encodes the redundancy needed for error-tolerant reconstruction. The primitives are designed so as to be "secure" even when an adversary learns the value of this public string.

Unfortunately, although these primitives suffice to obtain security in the presence of an eavesdropping adversary who learns pub as it is sent to the user, the work of Dodis *et al.* does not address the issue of malicious modification of pub. As a consequence, their work does not provide a method for secure authentication in the presence of an *active* adversary who may modify the messages sent between the server and the user. Indeed, depending on the specific sketch or fuzzy extractor being utilized, an adversary who maliciously alters the public string sent to a user may be able to learn that user's biometric data in its entirety. A "solution" is for the user to store pub himself rather than obtain it from the server (or to authenticate pub using a certificate chain), but this defeats the purpose of using biometric data in the first place: namely, to avoid the need for the user to store *any* additional cryptographic information — even if that information need not be kept secret.

Boyen [5], *inter alia*, partially addresses potential adversarial modification of pub (although his work focuses primarily on the orthogonal issue of re-using biometric data with multiple servers, which we do not explicitly address here). The main drawback of his technique in our context is that it provides only *unidirectional* authentication from the user to the server. Indeed, Boyen's approach cannot be used to achieve authentication of the server to the user since his definition of "insider security" (cf. [5–Section 5.2]) does not preclude an adversary from knowing the (incorrect) value s' of the shared secret recovered by the user when the adversary forwards a specially crafted pub' to this user; if the adversary knows s', then from the viewpoint of the user the adversary can do anything the

server could do, and hence authentication of the server to the user is impossible. The lack of mutual authentication implies that — when communicating over an insecure network — the user and server cannot securely establish a shared session key with which to encrypt and authenticate future messages: the user may unwittingly share a key with an adversary who can then decrypt any data sent by that user as well as authenticate arbitrary data.

1.1 Our Contributions

In this paper, we provide the first full solution to the problem of secure remote authentication using biometric data[1]: in particular, we show how to achieve mutual authentication and/or authenticated key exchange over a completely insecure channel. We offer two constructions. The first one is a generic solution which protects against modification of the public value pub in any context in which secure sketches or fuzzy extractors are used; thus, this solution serves as a drop-in replacement that "compiles" any protocol which is secure when pub is assumed to be transmitted reliably into one which is secure even when pub might be tampered with (we do not formalize this notion of "compilation", but rather view it as an intuitive way to understand our results). Our second construction is specific to the settings of remote authentication and key exchange, where it offers some improvements to the generic solution.

Compared with the work of Boyen [5], which was mostly concerned with the re-usability of biometrics, our constructions enjoy the following key advantages:

- Both of our solutions tolerate a stronger class of errors. In particular, Boyen's work only allows for *data-independent* errors, whereas our analysis handles *arbitrary* (but bounded) errors. We remark that small yet data-dependent errors seem natural in the context of biometric data.
- Our second solution is proven secure in the standard model.
- Our second solution achieves improved bounds on the entropy loss, on the order of 128 bits of entropy for practical choices of the parameters. This point is particularly important since the entropy of certain biometric features is roughly this order of magnitude (e.g., 173–250 bits for an iris scan [8, 13]).

Organization. We review some basic definitions as well as the sketches/fuzzy extractors of Dodis *et al.* [9] in Section 2. In Section 3 we introduce the notion of *robust* sketches/fuzzy extractors which are resilient to modification of the public value, and can be used as a generic replacement for sketches/fuzzy extractors in any application. Our second solution, which is specific to the problem of using biometric data for authentication and offers some advantages with respect to our generic construction, is described in Section 4.

[1] Of course, our techniques are applicable to *any* scenario which relies on secret data that, like biometric data, are non-uniform and/or not exactly reproducible.

2 Definitions

Unless explicitly stated otherwise, all logarithms are base 2. We let U_ℓ denote the uniform distribution over ℓ-bit strings. A *metric space* $(\mathcal{M}, d)$ is a finite set $\mathcal{M}$ equipped with a symmetric distance function $d : \mathcal{M} \times \mathcal{M} \to \mathbb{Z}^+ \cup \{0\}$ satisfying the triangle inequality and such that $d(x, y) = 0 \Leftrightarrow x = y$. (All metric spaces considered in this work are discrete, and the distances integer-valued.) For our application, we assume that the format of the biometric data is such that it forms a metric space under some appropriate distance function. We will not need to specify any particular metric space in our work, as our results build in a generic way on earlier sketch and fuzzy extractor constructions over any such space (e.g., those constructed in [9] for a variety of metrics).

A *probability space* (Ω, P) is a finite set Ω and a function $P : \Omega \to [0, 1]$ such that $\sum_{\omega \in \Omega} P(\omega) = 1$. A *random variable* W defined over the probability space (Ω, P) and taking values in a set $\mathcal{M}$ is a function $W : \Omega \to \mathcal{M}$. If (Ω, P) is a probability space over which two random variables W and W' taking values in a metric space $(\mathcal{M}, d)$ are defined, then we say that $d(W, W') \le t$ if for all $\omega \in \Omega$ it holds that $d(W(\omega), W'(\omega)) \le t$.

Given a metric space $(\mathcal{M}, d)$ and a point $x \in \mathcal{M}$ we define

$$\mathsf{Vol}_t^{\mathcal{M}}(x) \stackrel{\text{def}}{=} |\{x' \in \mathcal{M} \mid d(x, x') \le t\}|, \qquad \mathsf{Vol}_t^{\mathcal{M}} \stackrel{\text{def}}{=} \max_{x \in \mathcal{M}} \{\mathsf{Vol}_t^{\mathcal{M}}(x)\}.$$

The latter is the maximum number of points in any "ball" of radius t in $(\mathcal{M}, d)$.

Following [9], for a pair of random variables A and B, we define the *min-entropy* $H_\infty(A)$ of A, and the *average min-entropy of A given B*, as

$$H_\infty(A) = -\log(\max_a \Pr[A = a]), \quad \bar{H}_\infty(A|B) \stackrel{\text{def}}{=} -\log(\mathrm{Exp}_{b \leftarrow B}[2^{-H_\infty(A|B=b)}]).$$

The *statistical difference* between random variables A and B taking values in the same set D is defined as $\mathbf{SD}(A, B) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{d \in D} |\Pr[A = d] - \Pr[B = d]|$.

2.1 Secure Sketches and Fuzzy Extractors

We review the definitions from [9] using slightly different terminology. Recall from the introduction that a secure sketch provides a way to recover a shared secret w from any value w' which is a "close" approximation of w. More formally:

Definition 1. *An (m, m', t)-*secure sketch *over a metric space $(\mathcal{M}, d)$ comprises a* sketching procedure $\mathsf{SS} : \mathcal{M} \to \{0, 1\}^*$ *and a* recovery procedure Rec*, where:*

(Security) *For all random variables W over $\mathcal{M}$ such that $H_\infty(W) \ge m$, we have $\bar{H}_\infty(W \mid \mathsf{SS}(W)) \ge m'$.*

(Error tolerance) *For all pairs of points $w, w' \in \mathcal{M}$ with $d(w, w') \le t$, it holds that $\mathsf{Rec}(w', \mathsf{SS}(w)) = w$.* ♢

While secure sketches address the issue of error correction, they do not address the issue of the possible non-uniformity of W. Fuzzy extractors, defined next, correct for this.

Definition 2. *An* (m, ℓ, t, δ)-fuzzy extractor *over a metric space* $(\mathcal{M}, d)$ *comprises a (randomized)* extraction algorithm $\mathsf{Ext} : \mathcal{M} \to \{0,1\}^\ell \times \{0,1\}^*$ *and a* recovery procedure Rec *such that:*

(Security) *For all random variables* W *over* $\mathcal{M}$ *that satisfy* $H_\infty(W) \geq m$*, if* $\langle R, \mathsf{pub}\rangle \leftarrow \mathsf{Ext}(W)$ *then* $\mathbf{SD}(\langle R, \mathsf{pub}\rangle, \langle U_\ell, \mathsf{pub}\rangle) \leq \delta$.

(Error tolerance) *For all pairs of points* $w, w' \in \mathcal{M}$ *with* $d(w, w') \leq t$*, if* $\langle R, \mathsf{pub}\rangle \leftarrow \mathsf{Ext}(w)$ *then it is the case that* $\mathsf{Rec}(w', \mathsf{pub}) = R$. ♢

As shown in [9–Lemma 3.1], it is easy to construct a fuzzy extractor over a metric space $(\mathcal{M}, d)$ given any secure sketch defined over the same space, by applying a strong extractor [20] using a random "key" which is then included as part of pub. Starting with an (m, m', t)-secure sketch and with an appropriate choice of extractor, this transformation yields an $(m, m' - 2\log(\frac{1}{\delta}), t, \delta)$-fuzzy extractor.

2.2 Modeling Error in Biometric Applications

As error correction is a key motivation for our work, it is necessary to develop some formal model of the types of errors that may occur. In prior work by Boyen [5], the error in various biometric readings was assumed to be under adversarial control, with the restriction that the adversary could only specify data-*independent* errors (e.g., constant shifts, permutations, etc.). It is not clear that this is a realistic model in practice, as one certainly expects, say, portions of the biometric data where "features" are present to be more susceptible to error.

Here, we consider a much more general error model where the errors may be data-*dependent* and hence correlated not only with each other but also with the biometric secret itself. Furthermore, as we are ultimately interested in modeling "nature" — as manifested in the physical processes that cause fluctuations in the biometric measurements — we do not even require that the errors be efficiently computable (although we will impose this requirement in Section 4). The only restriction we make is that the errors be "small" and, in particular, less than the desired error-correction bound; since the error-correction bound in any real-world application should be selected to ensure correctness with high probability, this restriction seems reasonable. Formally:

Definition 3. *A* t-bounded distortion ensemble $\mathcal{W} = \{W_i\}_{i=0,\ldots}$ *is a sequence of random variables* $W_i : \Omega \to \mathcal{M}$ *such that for all* i *we have* $d(W_0, W_i) \leq t$. ♢

For our application, W_0 represents the biometric reading obtained when a user initially registers with a server, and W_i represents the biometric reading on the i^{th} authentication attempt by this user. Note that, regardless of the protocol used, an adversary can always impersonate the server if the adversary can guess W_i for some $i > 0$. The following lemmas give bound the probability of this occurrence. First, we show that the min-entropy of each W_i is, at worst, $\log(\mathsf{Vol}_t^{\mathcal{M}})$ bits less than that of W_0. Moreover, we show that W_i is no easier to guess than W_0 when $\mathsf{SS}(W_0)$ is available.

Lemma 1. *Let W_0, W_1 be random variables over $\mathcal{M}$ satisfying $d(W_0, W_1) \leq t$, and let B be an arbitrary random variable. Then*

$$\bar{H}_\infty(W_1 \mid B) \geq \bar{H}_\infty(W_0 \mid B) - \log \mathsf{Vol}_t^{\mathcal{M}}.$$

Proof. Fix $x \in \mathcal{M}$ and any outcome $B = b$. Since $d(W_0, W_1) \leq t$, we have $\Pr[W_1 = x \mid B = b] \leq \sum_{x' \mid d(x,x') \leq t} \Pr[W_0 = x' \mid B = b] \leq \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-H_\infty(W_0 \mid B=b)}$, which means that $H_\infty(W_1 \mid B = b) \geq H_\infty(W_0 \mid B = b) - \log \mathsf{Vol}_t^{\mathcal{M}}$. Since this relation holds for every b, the lemma follows. □

Secure sketches imply the following, stronger form of Lemma 1 which essentially states that points close to W_0 cannot be easier to guess than W_0 if the value of the sketch $\mathsf{SS}(W_0)$ is known.

Lemma 2. *Let W_0, W_1 be random variables over $\mathcal{M}$ satisfying $d(W_0, W_1) \leq t$, and let B be an arbitrary random variable. Let $(\mathsf{SS}, \mathsf{Rec})$ be a $(\star, \star, t)$-secure sketch. Then*

$$\bar{H}_\infty(W_1 \mid \mathsf{SS}(W_0), B) \geq \bar{H}_\infty(W_0 \mid \mathsf{SS}(W_0), B).$$

Proof. Notice that since $d(W_0, W_1) \leq t$, we have $\mathsf{Rec}(W_1, \mathsf{SS}(W_0)) = W_0$, which means that if for some x, b, pub we have $\Pr(W_1 = x \mid \mathsf{SS}(W_0) = \mathsf{pub}, B = b) \geq \alpha$, then $\Pr(W_0 = \mathsf{Rec}(x, \mathsf{pub}) \mid \mathsf{SS}(W_0) = \mathsf{pub}, B = b) \geq \alpha$ as well. Since this holds for all x, b and pub, the lemma follows. □

The analogue of Lemma 2 for fuzzy extractors holds as well (with $\mathsf{SS}(W_0)$ replaced by pub).

3 Robust Sketches and Fuzzy Extractors

Recall that a secure sketch, informally speaking, takes a secret w and returns some value pub which allows the recovery of w given any "close" approximation w' of w. When pub is transmitted to a user over an insecure network, however, an adversary might modify pub in transit. In this section, we define the notion of a *robust* sketch which protects against this sort of attack in a very strong way: with high probability, the user will detect that pub has been modified and can thus immediately abort in this case. A robust fuzzy extractor is defined similarly. We then show: (1) a construction of a robust sketch in the random oracle model, starting from any secure sketch; and (2) a conversion from any robust sketch to a robust fuzzy extractor; this conversion does not require random oracles. We conclude this section by showing the immediate application of robust fuzzy extractors to the problem of mutual authentication.

We first define a slightly stronger notion of a secure sketch:

Definition 4. *An (m, m', t)-secure sketch $(\mathsf{SS}, \mathsf{Rec})$ is said to be* well-formed *if it satisfies the conditions of Definition 1 except for the following modifications: (1) Rec may now return either an element in $\mathcal{M}$ or the distinguished symbol $\perp$; and (2) for all $w' \in \mathcal{M}$ and arbitrary pub', if $\mathsf{Rec}(w', \mathsf{pub}') \neq \perp$ then $d(w', \mathsf{Rec}(w', \mathsf{pub}')) \leq t$.* ♢

It is straightforward to transform any secure sketch $(\mathsf{SS}, \mathsf{Rec})$ into a well-formed secure sketch $(\mathsf{SS}, \mathsf{Rec}')$: Rec' runs Rec and then verifies that its output w is within distance t of the input w'. If yes, it outputs w; otherwise, it outputs $\perp$.

We now define the notion of a *robust* sketch:

Definition 5. *Given algorithms* $(\mathsf{SS}, \mathsf{Rec})$ *and random variables* $\mathcal{W} = \{W_0, W_1, \ldots, W_n\}$ *over metric space* $(\mathcal{M}, d)$*, consider the following game between an adversary* $\mathcal{A}$ *and a challenger: Let* w_0 *(resp.,* w_i*) be the value assumed by* W_0 *(resp.,* W_i*). The challenger computes* $\mathsf{pub} \leftarrow \mathsf{SS}(w_0)$ *and gives* pub *to* $\mathcal{A}$*. Next, for* $i = 1, \ldots, n$*, the adversary* $\mathcal{A}$ *outputs a "challenge"* $\mathsf{pub}_i \neq \mathsf{pub}$ *and is given* $\mathsf{Rec}(w_i, \mathsf{pub}_i)$ *in return. If there exists an* i *such that* $\mathsf{Rec}(w_i, \mathsf{pub}_i) \neq \perp$ *we say that the adversary* succeeds *and this event is denoted by* Succ*.*

We say that $(\mathsf{SS}, \mathsf{Rec})$ *is an* $(m, m'', n, \varepsilon, t)$-robust sketch *over* $(\mathcal{M}, d)$ *if it is a well-formed* $(m, \star, t)$*-secure sketch and: (1) for all* t*-bounded distortion ensembles* $\mathcal{W}$ *with* $H_\infty(W_0) \geq m$ *and all adversaries* $\mathcal{A}$ *we have* $\Pr[\mathsf{Succ}] \leq \varepsilon$*; and (2) the average min-entropy of* W_0*, conditioned on the entire view of* $\mathcal{A}$ *throughout the above game, is at least* m''*.*[2] ♢

A simpler definition would be to consider only random variables $\{W_0, W_1\}$ and to have $\mathcal{A}$ only output a single value $\mathsf{pub}_1 \neq \mathsf{pub}$. A standard hybrid argument would then imply the above definition with ε increased by a multiplicative factor of n. We have chosen to work with the more general definition above as it potentially allows for a tighter concrete security analysis. Also, although the above definition allows all-powerful adversaries, we will consider adversaries whose queries to a random oracle are bounded (but which are otherwise computationally unbounded). We remark that for a truly unbounded adversary (i.e., where even the oracle queries — if any — are unbounded), it is necessarily the case that $m'' \geq \log \frac{1}{\varepsilon}$ since at the last step of the game the adversary can guess W_n with probability $2^{-m''}$ and thus succeed with probability $\varepsilon \geq 2^{-m''}$.

3.1 Constructing a Generic Robust Sketch

Let $H : \{0,1\}^* \to \{0,1\}^k$ be a hash function. We construct a robust sketch $(\mathsf{SS}, \mathsf{Rec})$ from any well-formed secure sketch $(\mathsf{SS}^*, \mathsf{Rec}^*)$ as follows:

$\underline{\mathsf{SS}(w)}$	$\underline{\mathsf{Rec}(w, \mathsf{pub} = \langle \mathsf{pub}^*, h \rangle)}$
$\mathsf{pub}^* \leftarrow \mathsf{SS}^*(w)$	$w' = \mathsf{Rec}^*(w, \mathsf{pub}^*)$
$h = H(w, \mathsf{pub}^*)$	if $w' = \perp$ output $\perp$
return $\mathsf{pub} = \langle \mathsf{pub}^*, h \rangle$	if $H(w', \mathsf{pub}^*) \neq h$ output $\perp$
	otherwise, output w'

Theorem 1. *If* $(\mathsf{SS}^*, \mathsf{Rec}^*)$ *is a well-formed* (m, m', t)*-secure sketch over metric space* $(\mathcal{M}, d)$ *and* $H : \{0,1\}^* \to \{0,1\}^k$ *is a random oracle, then* $(\mathsf{SS}, \mathsf{Rec})$ *is an*

[2] In particular, this implies that $(\mathsf{SS}, \mathsf{Rec})$ is an (m, m'', t)-secure sketch.

$(m, m'', n, \varepsilon, t)$-robust sketch over $(\mathcal{M}, d)$ for any adversary making at most q_H queries to H, where

$$\varepsilon = (q_H^2 + n) \cdot 2^{-k} + (3q_H + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'},$$
$$m'' = -\log \varepsilon.$$

When $k \geq m' + \log q_H$ (which can be enforced in practice), the above simplifies to $\varepsilon \leq (4q_H + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'}$ and $m'' \geq m' - \log(4q_H + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}})$.

Proof (Sketch). It is easy to see that $(\mathsf{SS}, \mathsf{Rec})$ is an $(m, \star, t)$-secure sketch and thus we only need to prove the latter two conditions of Definition 5. In order to provide intuition, the following proof is somewhat informal; however, the arguments given here can easily be formalized. Let $\mathsf{pub} = \langle \mathsf{pub}^*, h \rangle$ denote the value output by SS in an execution of the game described in Definition 5. Note that if $\mathcal{A}$ ever outputs $\mathsf{pub}_i = \langle \mathsf{pub}_i^*, h_i \rangle$ with $\mathsf{pub}_i^* = \mathsf{pub}^*$ then the response is always $\perp$, since then we must have $h_i \neq h$ and so Rec will output $\perp$. Thus, we simply assume that $\mathsf{pub}_i^* \neq \mathsf{pub}^*$.

Fix a t-bounded distortion ensemble $\{W_0, W_1, \ldots, W_n\}$ with $H_\infty(W_0) \geq m$. For any output $\mathsf{pub}_i = \langle \mathsf{pub}_i^*, h_i \rangle$ of $\mathcal{A}$, define the random variable $W_i' \stackrel{\text{def}}{=} \mathsf{Rec}^*(W_i, \mathsf{pub}_i^*)$. In order not to complicate notation, we define

$$H_\infty(W_i') \stackrel{\text{def}}{=} -\log \left(\max_{x \in \mathcal{M}} \Pr[W_i' = x] \right);$$

i.e., we ignore the probability that $W_i' = \perp$ since $\mathcal{A}$ does not succeed in this case. $\bar{H}_\infty(W_i' \mid X)$ for a random variable X is defined similarly. Let w_0, w_i, and w_i' denote the values taken by the random variables W_0, W_i, and W_i', respectively.

We classify the random oracle queries of $\mathcal{A}$ into two types: *type 1* queries are those of the form $H(\cdot, \mathsf{pub}^*)$, and *type 2* queries are all the others. Informally, type 1 queries represent attempts by $\mathcal{A}$ to learn the value of w_0; in particular, if $\mathcal{A}$ finds w such that $H(w, \mathsf{pub}^*) = h$ then it is "likely" that $w_0 = w$. Type 2 queries represent attempts by $\mathcal{A}$ to determine an appropriate value for some h_i; i.e., if $\mathcal{A}$ "guesses" that $w_i' = w$ for a particular choice of pub_i^* then a "winning" strategy is for $\mathcal{A}$ to obtain $h_i = H(w, \mathsf{pub}_i^*)$ and output $\mathsf{pub}_i = \langle \mathsf{pub}_i^*, h_i \rangle$.

Without loss of generality, we assume that $\mathcal{A}$ makes all its type 1 queries first, then makes all its type 2 queries, and finishes by making its n queries to the challenger (cf. Definition 5) *in parallel* (i.e., non-adaptively). The validity of the assumption on the ordering of the type 1 and type 2 queries follows essentially from the analysis that follows. The assumption that all queries to the random oracle are made before any queries to the challenger is justified by the observation that if $\mathsf{Rec}(W_i, \mathsf{pub}_i) \neq \perp$ then the adversary has already succeeded, in which case we can end the game, whereas the only remaining response $\mathsf{Rec}(W_i, \mathsf{pub}_i) = \perp$ can be simulated by the adversary itself. This also justifies why we may assume that the adversary's challenges are made in parallel.

Let $\mathcal{Q}_1$ (resp., $\mathcal{Q}_2$) be a random variable denoting the sequence of type 1 (resp., type 2) queries made by $\mathcal{A}$ and the corresponding responses, and let q_1 (resp., q_2) denote the value assumed by Q_1 (resp., Q_2). For some fixed value of

pub, define $\gamma_{\mathsf{pub}} \stackrel{\text{def}}{=} H_\infty(W_0|\mathsf{pub})$. Notice, since $(\mathsf{SS}^*, \mathsf{Rec}^*)$ is an (m, m', t)-secure sketch, we have $\mathrm{Exp}_{\mathsf{pub}}[2^{-\gamma_{\mathsf{pub}}}] \leq 2^{-m'}$. Now, define $\gamma'_{\mathsf{pub},q_1} \stackrel{\text{def}}{=} H_\infty(W_0 \mid \mathsf{pub}, q_1)$, and let us call the value q_1 "bad" if $\gamma'_{\mathsf{pub},q_1} \leq \gamma_{\mathsf{pub}} - 1$. We consider two cases: If $2^{\gamma_{\mathsf{pub}}} \leq 2q_H$ we will not have any guarantees, but using Markov's inequality we have $\Pr[2^{\gamma_{\mathsf{pub}}} \leq 2q_H] = \Pr[2^{-\gamma_{\mathsf{pub}}} \geq 2^{-m'} \cdot (2^{m'}/2q_H)] \leq 2q_H \cdot 2^{-m'}$. Otherwise, if $2^{\gamma_{\mathsf{pub}}} > 2q_H$, we observe that the type 1 queries of $\mathcal{A}$ may be viewed as guesses of w_0. In fact, it is easy to see that we only improve the success probability of $\mathcal{A}$ if in response to a type 1 query of the form $H(w, \mathsf{pub}^*)$ we simply tell $\mathcal{A}$ whether $w_0 = w$ or not.[3] It is immediate that $\mathcal{A}$ learns the correct value of w_0 with probability at most $q_H \cdot 2^{-\gamma_{\mathsf{pub}}}$. Moreover, when this does *not* happen, $\mathcal{A}$ has eliminated at most $q_H \leq 2^{\gamma_{\mathsf{pub}}}/2$ (out of at least $2^{\gamma_{\mathsf{pub}}}$) possibilities for w_0, which means that $\gamma'_{\mathsf{pub},q_1} \geq \gamma_{\mathsf{pub}} - 1$, or in other words that q_1 is "good". Therefore, the probability that q_1 is "bad" in this second case is at most $q_H \cdot 2^{-\gamma_{\mathsf{pub}}}$.

Combining the above two arguments, we see that

$$\begin{aligned} \mathrm{Exp}_{\mathsf{pub}}[\Pr[q_1 \text{ bad}]] &\leq \Pr_{\mathsf{pub}}[2^{\gamma_{\mathsf{pub}}} \leq 2q_H] + \mathrm{Exp}_{\mathsf{pub}}[q_H \cdot 2^{-\gamma_{\mathsf{pub}}}] \\ &\leq 2q_H \cdot 2^{-m'} + q_H \cdot 2^{-m'} \quad = \quad 3q_H \cdot 2^{-m'}. \end{aligned} \tag{1}$$

Next, define $\gamma''_{\mathsf{pub},q_1} \stackrel{\text{def}}{=} \min_i(H_\infty(W'_i \mid \mathsf{pub}, q_1))$. Recall that $\{W_0, W_1, \ldots\}$ is a t-bounded distortion ensemble which means $d(W_0, W_i) \leq t$. Furthermore, since $(\mathsf{SS}^*, \mathsf{Rec}^*)$ is well-formed, $\{W_i, W'_i\}$ is also a t-bounded distortion ensemble[4] regardless of pub^*_i, which means $d(W_i, W'_i) \leq t$. Applying Lemma 2 on $\{W_0, W_i\}$ (noticing that pub contains pub^*), followed by Lemma 1 on $\{W_i, W'_i\}$, we have

$$\gamma''_{\mathsf{pub},q_1} \geq \min_i(H_\infty(W_i \mid \mathsf{pub}, q_1)) - \log \mathsf{Vol}_t^{\mathcal{M}} \quad \geq \quad \gamma'_{\mathsf{pub},q_1} - \log \mathsf{Vol}_t^{\mathcal{M}}. \tag{2}$$

We now consider the type 2 queries made by $\mathcal{A}$. Clearly, the answers to these queries do not affect the conditional min-entropies of W'_i (since these queries do not include pub^*), so the best probability for the attacker to predict any of the W'_i is still given by $2^{-\gamma''_{\mathsf{pub},q_1}}$, for fixed pub and q_1. Assume for a moment that there are no collisions in the outputs of any of the adversary's random oracle queries, and consider the adversary's i^{th} query $\langle \mathsf{pub}^*_i, h_i \rangle$ to the challenger. The probability that this query is "successful" is at most the probability that $\mathcal{A}$ asked a type 2 query of the form $H(w'_i, \cdot)$ for the correct w'_i plus the probability that such a query was not asked, yet $\mathcal{A}$ nevertheless managed to predict the value $H(w'_i, \mathsf{pub}^*_i)$. Clearly, the second case happens with probability at most 2^{-k}. As for the first case, for any h_i there is at most one w for which $H(w, \cdot) = h_i$, since, by assumption, there are no collisions in these type 2 queries. Thus, the adversary succeeds on its i^{th} query if this w is equal to the correct value w'_i. By what we just argued, the probability that this occurs is at most $2^{-\gamma''_{\mathsf{pub},q_1}}$, irrespective of pub^*_i.

[3] This has no effect when $H(w, \mathsf{pub}^*) \neq h$ as then $\mathcal{A}$ learns anyway that $w \neq w_0$. The modification has a small (but positive) effect on the success probability of $\mathcal{A}$ when $H(w, \mathsf{pub}^*) = h$ since this fact by itself does not definitively guarantee that $w = w_0$.

[4] This ignores the case when $W'_i = \perp$; see the definition of $H_\infty(W'_i)$ given earlier.

Therefore, assuming no collisions in type 2 queries, the success probability of $\mathcal{A}$ in any one of its n parallel queries is at most $n \cdot (2^{-\gamma''_{\mathsf{pub},q_1}} + 2^{-k})$. Furthermore, by the birthday bound the probability of a collision is at most $q_H^2/2^k$. Therefore, conditioned on pub and q_1 and for the corresponding value of $\gamma''_{\mathsf{pub},q_1}$, we find that $\Pr[\mathsf{Succ} \mid \mathsf{pub}, q_1] \leq n \cdot 2^{-\gamma''_{\mathsf{pub},q_1}} + (q_H^2 + n) \cdot 2^{-k}$.

To conclude, the adversary's overall probability of success is thus bounded by the expectation, over pub and q_1, of this previous quantity; that is:

$$\begin{aligned}\Pr[\mathsf{Succ}] &= \mathrm{Exp}_{\mathsf{pub},q_1}\left[\Pr[\mathsf{Succ} \mid \mathsf{pub}, q_1]\right] \\ &\leq (q_H^2 + n) \cdot 2^{-k} \\ &\quad + \mathrm{Exp}_{\mathsf{pub}}\left[\Pr_{q_1 \leftarrow \mathcal{Q}_1}[q_1 \text{ bad} \mid \mathsf{pub}] + \sum_{q_1 \text{ good}} n \cdot 2^{-\gamma''_{\mathsf{pub},q_1}} \cdot \Pr[Q_1 = q_1 \mid \mathsf{pub}]\right].\end{aligned}$$

Using Equation 2, we see that $2^{-\gamma''_{\mathsf{pub},q_1}} \leq \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-\gamma'_{\mathsf{pub},q_1}}$. Moreover, for good q_1 we have $\gamma'_{\mathsf{pub},q_1} \geq \gamma_{\mathsf{pub}} - 1$, which means that $2^{-\gamma''_{\mathsf{pub},q_1}} \leq 2 \cdot \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-\gamma_{\mathsf{pub}}}$. Finally, using Equation 1, we have $\mathrm{Exp}_{\mathsf{pub}}[\Pr[q_1 \text{ bad} \mid \mathsf{pub}]] \leq 3q_H \cdot 2^{-m'}$. Combining all these, we successively derive:

$$\begin{aligned}\Pr[\mathsf{Succ}] &\leq (q_H^2 + n) \cdot 2^{-k} + 3q_H \cdot 2^{-m'} \\ &\quad + \mathrm{Exp}_{\mathsf{pub}}\left[2n \cdot \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-\gamma_{\mathsf{pub}}} \cdot \Pr_{q_1 \leftarrow \mathcal{Q}_1}[q_1 \text{ good}]\right] \\ &\leq (q_H^2 + n) \cdot 2^{-k} + 3q_H \cdot 2^{-m'} + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}} \cdot \mathrm{Exp}_{\mathsf{pub}}\left[2^{-\gamma_{\mathsf{pub}}}\right] \\ &\leq (q_H^2 + n) \cdot 2^{-k} + (3q_H + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'} \;=\; \varepsilon.\end{aligned}$$

As for the claimed value of m'', we omit the details since they follow almost the same argument. As above, assuming that q_1 is good, that no collisions occur in type 2 queries, and that the adversary does not manage to guess any of the values $H(w'_i, \mathsf{pub}^*_i)$, the conditional min-entropy of W_0 is at least $\gamma'_{\mathsf{pub},q_1} - \log(n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \geq \gamma_{\mathsf{pub}} - 1 - \log(n \cdot \mathsf{Vol}_t^{\mathcal{M}})$. On the other hand, all these bad events leading to a possibly smaller min-entropy of W_0 happen with (expected) probability (over pub) at most $(q_H^2 + n) \cdot 2^{-k} + 3q_H \cdot 2^{-m'}$. From this, it is easy to see that if View represents the adversary's view in the experiment, then

$$\begin{aligned}\bar{H}_\infty(W_0 | \mathsf{View}) &\geq -\log\Big((q_H^2 + n) \cdot 2^{-k} + 3q_H \cdot 2^{-m'} \\ &\qquad + \mathrm{Exp}_{\mathsf{pub}}\left[2n \cdot \mathsf{Vol}_t^{\mathcal{M}} \cdot 2^{-\gamma_{\mathsf{pub}}}\right]\Big) \\ &\geq -\log\left((q_H^2 + n) \cdot 2^{-k} + (3q_H + 2n \cdot \mathsf{Vol}_t^{\mathcal{M}}) \cdot 2^{-m'}\right) \;=\; m''.\end{aligned} \tag{3}$$

□

We remark that the above proof uses only a non-programmable random oracle.

The bound on ε that we derive in the above proof has an intuitive interpretation. The sub-expression $\left(q_H + n \cdot \mathsf{Vol}_t^{\mathcal{M}}\right) \cdot 2^{-m'}$ that appears (up to a small

constant factor due to the analysis) can be viewed as the probability that the adversary "gets information" about the point w_0: The contribution $q_H \cdot 2^{-m'}$ is due to the type 1 oracle queries where, for each of at most q_H queries, the adversary "hits" the correct value of w_0 with probability $2^{-m'}$. Then, each of the adversary's n challenges cover no more than $\mathsf{Vol}_t^{\mathcal{M}}$ candidates for w_0, since each such query eliminates at most one value for w_i' (unless collisions in type 2 queries occur), which in turn eliminates up to $\mathsf{Vol}_t^{\mathcal{M}}$ candidates for w_i, each of which can only eliminate one candidate $\mathsf{Rec}(w_i, \mathsf{pub}^*)$ for w_0. Besides the above, the other contributions to ε are due to the probability of collisions in the random oracle, plus a small term to account for the possibility that the adversary can guess the output of the random oracle at an unqueried point.

In practice, one can set k large enough so that $\max(q_H,\ n \cdot \mathsf{Vol}_t^{\mathcal{M}})$ is the dominant factor determining the amount of the additional "loss" incurred as compared to regular "non-robust" sketches.

3.2 From Robust Sketches to Robust Fuzzy Extractors

In a manner exactly analogous to the above, we may define the notion of a robust fuzzy extractor. We include the definition here since we refer to it in the next subsection:

Definition 6. *Given algorithms* $(\mathsf{Ext}, \mathsf{Rec})$ *and random variables* $\mathcal{W} = \{W_0, W_1, \ldots, W_n\}$ *over a metric space* $(\mathcal{M}, d)$*, consider the following game between an adversary* $\mathcal{A}$ *and a challenger: Let* w_0 *(resp.,* w_i*) be the value assumed by* W_0 *(resp.,* W_i*). The challenger computes* $(R, \mathsf{pub}) \leftarrow \mathsf{Ext}(w_0)$ *and gives* pub *to* $\mathcal{A}$*. Next, for* $i = 1, \ldots, n$*, the adversary* $\mathcal{A}$ *outputs* $\mathsf{pub}_i \neq \mathsf{pub}$ *and is given* $\mathsf{Rec}(w_i, \mathsf{pub}_i)$ *in return. If there exists an* i *such that* $\mathsf{Rec}(w_i, \mathsf{pub}_i) \neq \perp$*, we say the adversary* succeeds *and this event is denoted by* Succ*.*

We say $(\mathsf{Ext}, \mathsf{Rec})$ *is an* $(m, \ell, n, \varepsilon, t, \delta)$-robust fuzzy extractor *over* $(\mathcal{M}, d)$ *if the following hold for all* t*-bounded distortion ensembles* $\mathcal{W}$ *with* $H_\infty(W_0) \geq m$*:*

(Robustness) *For all adversaries* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{Succ}] \leq \varepsilon$*.*
(Security) *Let* View *denote the entire view of* $\mathcal{A}$ *at the conclusion of the above game. Then,* $\mathbf{SD}(\langle R, \mathsf{View}\rangle, \langle U_\ell, \mathsf{View}\rangle) \leq \delta$*.*
(Error-tolerance) *For all* w' *with* $d(w_0, w') \leq t$*, we have* $\mathsf{Rec}(w', \mathsf{pub}) = R$*.* $\diamondsuit$

By applying a pairwise-independent hash function (i.e., a strong extractor) almost exactly as in [9–Lemma 3.1], we can convert any robust sketch to a robust fuzzy extractor in the standard model (losing, as there, $2 \log \delta^{-1}$ bits of entropy). A subtlety is that we need to "bind" the hash function key to the sketch itself. We do this using a primitive we call a *labeled* robust sketch which, in a nutshell, defines a family of robust sketches indexed by a label. (For example, in the specific construction of robust sketches from the previous section labels can be incorporated by including the label as the input to the hash function which is modeled as a random oracle.) By using the key to the hash function as a label, we can bind the key to the sketch as needed. Details will appear in the full version.

We remark that if we are content to assume a random oracle G (as we anyway only know how to construct robust sketches in the random oracle model), we can trivially "extract" from a random variable w by computing $G(w)$. This has the advantage of not losing $2\log\delta^{-1}$ bits of entropy when extracting. In this case, we achieve $\delta \leq q_G \cdot 2^{-m''}$, where q_G is the number of queries to G and m'' is as in Theorem 1.

3.3 Application to Secure Authentication

The application of a robust fuzzy extractor to achieve mutual authentication or authenticated key exchange over an insecure channel is immediate. Given any secure protocol Π (say, for authenticated key exchange) based on a uniformly-distributed shared key of length ℓ, any $(m, \ell, n, \varepsilon, t, \delta)$-robust fuzzy extractor $(\mathsf{Ext}, \mathsf{Rec})$, and any source W_0 with $H_\infty(W_0) \geq m$, consider the protocol Π' constructed as follows:

Initialization. The user samples w_0 according to W_0 (i.e., takes a scan of his biometric data) and computes $(R, \mathsf{pub}) \leftarrow \mathsf{Ext}(w_0)$. The user registers (R, pub) at the server.

Protocol execution. The i^{th} time the user wants to run the protocol, the user will sample w_i according to some distribution W_i (i.e., the user re-scans his biometric data). The server sends pub to the user, who then computes $\hat{R} = \mathsf{Ext}(w_i, \mathsf{pub})$. If $\hat{R} = \perp$, the user immediately aborts. Otherwise, the server and user execute protocol Π, with the server and the user respectively using the keys R and $\hat{R}$.

Assume that $\mathcal{W} = \{W_0, W_1, \ldots\}$ is a t-bounded distortion ensemble. Correctness of the above protocol is easily seen to hold: if the user obtains the correct value of pub from the server then, because $d(w_0, w_i) \leq t$, the user will recover $\hat{R} = R$ and thus both user and server will end up using the same key R in the underlying protocol Π. The security of Π' with respect to the definitions of [3, 1], which consider an active adversary who may control all messages sent between the user and the server, follows from the following observations:

- If the adversary forwards $\mathsf{pub}' \neq \mathsf{pub}$ to at most n different user-instances, these instances will all abort immediately (without running Π) except with probability at most ε. Thus, except with this probability, the adversary is limited to forwarding the correct value of pub.
- When the adversary forwards pub unchanged, the user and server run an execution of Π using a key R which is within statistical difference δ from a uniformly distributed ℓ-bit key. Note that this is true even when conditioned on the view of the adversary in sessions when it does *not* forward pub unchanged (cf. Definition 6). Thus, assuming Π is secure, the adversary will not succeed in "breaking" Π' in this case either.

In terms of concrete security, if the security of Π against an adversary who executes at most n sessions with the user and the server is ε_Π, then the security

of Π' is $\varepsilon+\delta+\varepsilon_\Pi$. A formal proof following the above intuition is straightforward, and will appear in the full version of this work.

4 Improved Solution Tailored for Mutual Authentication

As discussed in the introduction, the robust sketches and fuzzy extractors described in the previous section provide a general mechanism for dealing with adversarial modification of the public value pub. In particular, taking any protocol based on the secure sketches or fuzzy extractors of [9] which is secure when the public value is assumed *not* to be tampered with, and plugging in a *robust* sketch or fuzzy extractor, yields a protocol secure against an adversary who may either modify the contents of the server — as in the case where the server itself is malicious — or else modify the value of pub when it is sent to the user.

For specific problems of interest, however, it remains important to explore solutions which might improve upon the general-purpose solution described above. In this section, we show that for the case of mutual authentication and/or authenticated key exchange an improved solution is indeed possible. As compared to the generic solution based on robust fuzzy extractors (cf. Section 3.3), the solution described here has the advantages that: (1) it is provably secure in the standard model; and (2) it can achieve improved bounds on the "effective entropy loss". We provide an overview of our solution now.

Given the proof of Theorem 1, the intuition behind our current solution is actually quite straightforward. As in that proof, let $\mathcal{W} = \{W_0, \ldots\}$ be a sequence of random variables where W_0 represents the initial recorded value of the user's biometric data and W_i denotes the i^{th} scanned value of the biometric data. Given a well-formed secure sketch $(\mathsf{SS}^*, \mathsf{Rec}^*)$ and a value $\mathsf{pub}_i^* \neq \mathsf{pub}^* = \mathsf{SS}^*(W_0)$ chosen by the adversary, let $W_i' \stackrel{\text{def}}{=} \mathsf{Rec}(W_i, \mathsf{pub}_i^*)$ and define the min-entropy of W_i' as in the proof of Theorem 1. At a high level, Theorem 1 follows from the observations that: (1) the average min-entropy of W_i' is "high" for *any* value pub_i^*; and (2) since the adversary succeeds only if it can also output a value $h_i = H(W_i', \mathsf{pub}_i^*)$, where H is a random oracle, the adversary is essentially unable to succeed with probability better than $2^{-H_\infty(W_i')}$ in the i^{th} iteration. Crucial to the proof also is the fact that, except with "small" probability, the value $h = H(W_0, \mathsf{pub}^*)$ does not reduce the entropy of W_0 "very much" (again using the fact that H is a random oracle).

The above suggests that another way to ensure that the adversary does not succeed with probability better than $2^{-H_\infty(W_i')}$ in any given iteration would be to have the user run an "equality test" using its recovered value W_i'. If this equality test is "secure" (in some appropriate sense we have not yet defined) then the adversary will effectively be reduced to simply guessing the value of W_i', and hence its success probability in that iteration will be as claimed. Since we have already noted that the average min-entropy of W_i' is "high" when any well-formed secure sketch is used (regardless of the value pub_i^* chosen by the adversary), this will be sufficient to ensure security of the protocol overall.

Thinking about what notion of security this "equality test" should satisfy, one realizes that it must be secure for arbitrary distributions on the user's secret value, and not just uniform ones. Also, the protocol must ensure that each interaction by the adversary corresponds to a guess of (at most) one possible value for W_i'. Finally, since the protocol is meant to be run over an insecure network, it must be "non-malleable" in some sense so that the adversary cannot execute a man-in-the-middle attack when the user and server are both executing the protocol. Finally, the adversary should not gain any information about the user's true secret W_0 (at least in a computational sense) after passively eavesdropping on multiple executions of the protocol. With the problem laid out in this way, it becomes clear that one possibility is to use a password-only authenticated key exchange (PAK) protocol [4, 1, 6] as the underlying "equality test".

Although the above intuition is appealing, we remark that a number of subtleties arise when trying to apply this idea to obtain a provably secure solution. In particular, we will require the PAK protocol to satisfy a slightly stronger definition of security than that usually considered for PAK (cf. [1, 6, 12]); informally, the PAK protocol should remain "secure" even when: (1) the adversary can dynamically add clients to the system, with (unique) identities chosen by the adversary; (2) the adversary can specify *non-uniform* and *dependent* password distributions for these clients; and (3) the adversary can specify such distributions *adaptively* at the time the client is added to the system. Luckily, it is not difficult to verify that at least some existing protocols (e.g., [1, 17, 18, 11, 16]) satisfy a definition of this sort.[5] (Interestingly, the recent definition of [7] seems to imply the above properties.) Due to lack of space, the formal definition of security required for our application is deferred to the full version.

4.1 A Direct Construction

With the above in mind, we now describe our construction. Let Π be a PAK protocol and let $(\mathsf{SS}, \mathsf{Rec})$ be a well-formed secure sketch. Construct a modified protocol Π' as follows:

Initialization. User U samples w_0 according to W_0 (i.e., takes a scan of his biometric data) and computes $\mathsf{pub} \leftarrow \mathsf{SS}(w_0)$. The user registers (w_0, pub) at the server S.

Protocol execution (server). The server sends pub to the user. It then executes protocol Π using the following parameters: it sets its own "identity" (within Π) to be $S\|\mathsf{pub}$, its "partner identity" to be $\mathsf{pid} = U\|\mathsf{pub}$, and the "password" to be w_0.

Protocol execution (user). The i^{th} time the user executes the protocol, the user first samples w_i according to distribution W_i (i.e., the user re-scans his biometric data). The user also obtains a value pub' in the initial message it

[5] In fact, it is already stated explicitly in [17, 11] that the given protocols remain secure even under conditions 1 and 2, and it is not hard to see that they remain secure under condition 3 as well.

receives, and computes $w' = \mathsf{Rec}(w_i, \mathsf{pub}')$. If $w' = \perp$ then the user simply aborts. Otherwise, the user executes protocol Π, setting its own "identity" to $U \| \mathsf{pub}'$, its "partner identity" to $S \| \mathsf{pub}'$, and using the "password" w'.

It is easy to see that correctness holds, since if the user and the server interact without any interference from the adversary then: (1) the identity used by the server is equal to the partner ID of the user; (2) the identity of the user is the same as the partner ID of the server; and (3) the passwords w_0 and w' are identical. Before discussing the security of this protocol, we need to introduce a slight restriction of the notion of a t-bounded distortion ensemble in which the various random variables in the ensemble are (efficiently) computable:

Definition 7. *Let $(\mathcal{M}, d)$ be a metric space. An* explicitly computable *t-bounded distortion ensemble is a sequence of boolean circuits $\mathcal{W} = \{W_0, \ldots\}$ and a parameter ℓ such that, for all i, the circuit W_i computes a function from $\{0,1\}^\ell$ to $\mathcal{M}$ and, furthermore, for all $r \in \{0,1\}^\ell$ we have $d(W_0(r), W_i(r)) \leq t$.* $\diamondsuit$

In our application, $\mathcal{W}$ will be output by a PPT adversary, ensuring both that the ensemble contains only a polynomial number of circuits and that each such circuit is of polynomial size (and hence may be evaluated efficiently). We remark that it is *not* necessary for our proof that it be possible to efficiently verify whether a given $\mathcal{W}$ satisfies the "t-bounded" property or whether the min-entropy of W_0 is as claimed, although the security guarantee stated below only holds if $\mathcal{W}$ does indeed satisfy these properties.[6] With the above in mind, we now state the security achieved by our protocol:

Theorem 2. *Let Π be a secure PAK protocol (with respect to the definition sketched earlier) and let $\mathcal{A}$ be a* PPT *adversary. If $(\mathsf{SS}, \mathsf{Rec})$ is a well-formed (m, m', t)-secure sketch over a metric space $(\mathcal{M}, d)$, and $\mathcal{W} = \{W_0, \ldots\}$ is an explicitly-computable t-bounded distortion ensemble (output adaptively by $\mathcal{A}$) with $H_\infty(W_0) \geq m$, then the success probability of $\mathcal{A}$ in attacking protocol Π' is at most $q_s \cdot 2^{-m''} + \mathsf{negl}(\kappa)$, where q_s represents the number of sessions in which the adversary attempts to impersonate one of the parties, and $m'' = m' - \log \mathsf{Vol}_t^{\mathcal{M}}$.*

Due to space limitations, the proof is deferred to the full version.

Specific instantiations. As noted earlier, a number of PAK protocols satisfying the required definition of security are known. If one is content to work in the random oracle model then the protocol of [1] may be used (note that this still represents an improvement over the solution based on robust fuzzy extractors since the "effective key size" will be larger, as we discuss in the next paragraph). To obtain a solution in the standard model which is only slightly less efficient, the

[6] As to whether the adversary can be "trusted" to output a $\mathcal{W}$ satisfying these properties, recall that $\mathcal{W}$ anyway is meant to model naturally-occurring errors. Clearly, if a real-world adversary has the ability to, e.g., introduce arbitrarily-large errors then only weaker security guarantees can be expected to hold.

PAK protocols of [17, 11, 16] could be used.[7] Note that although these protocols were designed for use with "short" passwords, they can be easily modified to handle "large" passwords without much loss of efficiency; we discuss this further in the full version.

4.2 Comparing Our Two Solutions

It is somewhat difficult to compare the security offered by our two solutions (i.e., the one based on robust fuzzy extractors and the one described in this section) since an exact comparison depends on a number of assumptions and design decisions. As we already observed, the main advantage of the solution described in this section is that it does not rely on random oracles. On the other hand, the solution based on robust fuzzy extractors is simpler and more efficient.

The solution presented in this section does not require any randomness extraction, and it therefore "saves" $2 \log \delta^{-1}$ bits of entropy as compared with solutions that apply standard randomness extractors to the recovered biometric data. Since a likely value in practice is $\delta \leq 2^{-64}$, this results in a potential savings of at least 128 bits of entropy. When the entropy of the original biometric data is "large", however, we notice that (1) as mentioned already in the previous section, we may use a random oracle as our randomness extractor and thereby avoid the loss of $2 \log \delta^{-1}$ bits of entropy; and (2) our two approaches can be combined, and one can use a PAK protocol with any *robust* sketch. If this is done then additional extraction is not required, and so we again avoid losing $2 \log \delta^{-1}$ bits of entropy.

On the other hand, the solution of the present section offers a clear advantage when the entropy of the original biometric data is "small". Although in this case the adversary can succeed by an exhaustive, on-line "dictionary" attack, the security of our second solution implies that this is the *best* an adversary can do. In contrast, our solution based on robust sketches would not be appropriate in this case since the adversary could determine the user's secret biometric data using *off-line* queries to the random oracle (cf. the factor proportional to $q_H \cdot 2^{-m'}$ in Theorem 1).

References

1. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. *Adv. in Cryptology — Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 139–155, 2000.
2. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. ACM CCS 1993, ACM Press, 1993.
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *Adv. in Cryptology — Crypto 1993*, LNCS vol. 773, Springer-Verlag, pp. 232–249, 1993.

[7] Although these protocols require public parameters, such parameters can be "hard coded" into the implementation of the protocol and are fixed for all users of the system; thus, users are not required to remember or store these values. The difference is akin to the difference between PAK protocols in a "hybrid" PKI model (where clients store their server's public key) and PAK protocols (including [17, 11, 16]) in which clients need only remember a short password.

4. S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. *IEEE Symposium on Research in Security and Privacy*, IEEE, pp. 72–84, 1992.
5. X. Boyen. Reusable Cryptographic Fuzzy Extractors. ACM CCS 2004, ACM Press, pp. 82–91, 2004.
6. V. Boyko, P. MacKenzie, and S. Patel. Provably-Secure Password-Authenticated Key Exchange Using Diffie-Hellman. *Adv. in Cryptology — Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 156–171, 2000.
7. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie. Universally Composable Password-Based Key Exchange. Eurocrypt 2005 (these proceedings).
8. G. Davida, Y. Frankel, and B. Matt. On Enabling Secure Applications Through Off-Line Biometric Identification. IEEE Security and Privacy '98.
9. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *Adv. in Cryptology — Eurocrypt 2004*, LNCS vol. 3027, Springer-Verlag, pp. 523–540, 2004.
10. N. Frykholm and A. Juels. Error-Tolerant Password Recovery. ACM CCS 2001.
11. R. Gennaro and Y. Lindell. A Framework for Password-Based Authenticated Key Exchange. *Adv. in Cryptology — Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 524–543, 2003.
12. O. Goldreich and Y. Lindell. Session-Key Generation Using Human Passwords Only. *Adv. in Cryptology — Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 408–432, 2001.
13. A. Juels. Fuzzy Commitment. Slides from a presentation at DIMACS, 2004. Available at `http://dimacs.rutgers.edu/Workshops/Practice/slides/juels.ppt`
14. A. Juels and M. Sudan. A Fuzzy Vault Scheme. IEEE Intl. Symp. on Info. Theory, 2002.
15. A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. ACM CCS 1999, ACM Press, 1999.
16. J. Katz, P. MacKenzie, G. Taban, and V. Gligor. Two-Server Password-Only Authenticated Key Exchange. Manuscript, Jan. 2005.
17. J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. *Adv. in Cryptology — Eurocrypt 2001*, LNCS vol. 2045, Springer-Verlag, pp. 475–494, 2001.
18. J. Katz, R. Ostrovsky, and M. Yung. Forward Secrecy in Password-Only Key-Exchange Protocols. *Security in Communication Networks: SCN 2002*, LNCS vol. 2576, Springer-Verlag, pp. 29–44, 2002.
19. F. Monrose, M. Reiter, and S. Wetzel. Password Hardening Based on Keystroke Dynamics. ACM CCS 1999, ACM Press, 1999.
20. N. Nisan and A. Ta-Shma. Extracting Randomness: A Survey and New Constructions. J. Computer and System Sciences 58(1): 148–173, 1999.
21. P. Tuyls and J. Goseling. Capacity and Examples of Template-Protecting Biometric Authentication Systems. Biometric Authentication Workshop, 2004.
22. E. Verbitskiy, P. Tuyls, D. Denteneer, and J.-P. Linnartz. Reliable Biometric Authentication with Privacy Protection. 24th Benelux Symp. on Info. Theory, 2003.

Stronger Security Bounds for Wegman-Carter-Shoup Authenticators

Daniel J. Bernstein*

Department of Mathematics, Statistics, and Computer Science (M/C 249),
The University of Illinois at Chicago,
Chicago, IL 60607–7045
djb@cr.yp.to

Abstract. Shoup proved that various message-authentication codes of the form $(n, m) \mapsto h(m) + f(n)$ are secure against all attacks that see at most $\sqrt{1/\epsilon}$ authenticated messages. Here m is a message; n is a nonce chosen from a public group G; f is a secret uniform random permutation of G; h is a secret random function; and ϵ is a differential probability associated with h.

Shoup's result implies that if AES is secure then various state-of-the-art message-authentication codes of the form $(n, m) \mapsto h(m) + \mathrm{AES}_k(n)$ are secure up to $\sqrt{1/\epsilon}$ authenticated messages. Unfortunately, $\sqrt{1/\epsilon}$ is only about 2^{50} for some state-of-the-art systems, so Shoup's result provides no guarantees for long-term keys.

This paper proves that security of the same systems is retained up to $\sqrt{\#G}$ authenticated messages. In a typical state-of-the-art system, $\sqrt{\#G}$ is 2^{64}. The heart of the paper is a very general "one-sided" security theorem: $(n, m) \mapsto h(m) + f(n)$ is secure if there are small upper bounds on differential probabilities for h and on interpolation probabilities for f.

1 Introduction

This paper proves that various state-of-the-art 128-bit authenticators are secure against all attacks that see at most 2^{64} authenticated messages. Previous proofs broke down at a smaller number of messages, often below 2^{50}.

A typical example

Here is a well-known polynomial-evaluation message-authentication code over a field of size 2^{128}.

* The author was supported by the National Science Foundation under grant CCR–9983950, and by the Alfred P. Sloan Foundation. Date of this document: 2005.02.27. Permanent ID of this document: 2d603727f69542f30f7da2832240c1ad. This version is final and may be freely cited. Priority dates: The first version of this document was posted 2004.10.19.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 164–180, 2005.

Each message is a polynomial in one variable over the field. The polynomial's constant coefficient is required to be 0. One authenticates the polynomial by evaluating it at a point and adding a function of the message number: the sender's nth message, say m_n, is transmitted as $(n, m_n, m_n(r) + f(n))$. Here r and f are secrets shared by the sender and the receiver.

It is easy to prove information-theoretic security of this system if r and f are independent, r is a uniform random element of the field, and f is a uniform random function from $\{n\}$ to the field—in other words, if $r, f(1), f(2), \ldots$ are independent uniform random elements of the field. The attacker's chance of successfully forging a message is at most $LD/2^{128}$, where L is the maximum degree of a message and D is the number of forgeries attempted. The idea of the proof is that $m_n(r) + f(n)$ leaks no information about $m_n(r)$.

What if f is a uniform random *injective* function—in other words, what if $f(1), f(2), \ldots$ are chosen to be distinct? What is the attacker's chance of successfully forging a message? Here are three answers:

- The easy bound: Say the sender transmits only C messages, where C is small. Then $f(1), f(2), \ldots, f(C)$ are *nearly* independent, and one can easily prove that the attacker's chance of success is at most $LD/2^{128} + C(C-1)/2^{129}$. This bound becomes useless as C approaches 2^{64}.
- Another bound: Shoup proved in [19–Theorem 2] that the attacker's chance of success is at most $2LD/2^{128}$ if $C \leq 2^{64}/\sqrt{L}$.
- A better bound: This paper proves that the attacker's chance of success is below $1.002LD/2^{128}$ if $C \leq 2^{60}$, and below $1.7LD/2^{128}$ if $C \leq 2^{64}$, and below $3000LD/2^{128}$ if $C \leq 2^{66}$.

For example, say the sender authenticates $C = 2^{60}$ messages, the attacker tries $D = 2^{60}$ forgeries, and the maximum message degree is $L = 2^{16}$. The easy bound is about $1/2^9$, which is not at all comforting. Shoup's bound is inapplicable. The bound in this paper is $1.002/2^{52}$.

Consequences for AES-based authenticators

Despite the high speed and information-theoretic security of $m_n(r) + f(n)$, users often prefer $m_n(r) + \mathrm{AES}_k(n)$. Why? Because r, k occupy only 32 bytes, whereas $r, f(1), f(2), \ldots$ occupy an additional 16 bytes for each message.

Define δ as the attacker's chance of distinguishing AES_k from f, where f is a uniform random injective function on 16-byte blocks. The important feature of AES for this paper is that δ is *believed* to be extremely small, even after 2^{64} or more queries, even for an attacker with incredible computational resources. This was an explicit design goal of AES: [2–Section 4] identified "the extent to which the algorithm output is indistinguishable from [the output of] a [uniform] random permutation" as one of the "most important" factors in evaluating AES proposals, and [17–Table 1] shows that AES evaluators considered many attacks aimed at this feature.

The attacker's success chance against $m_n(r) + \mathrm{AES}_k(n)$ is at most δ plus the attacker's success chance against $m_n(r) + f(n)$. The point of this paper is

that the second chance is extremely small, even for $C = 2^{64}$. Consequently, $m_n(r) + \mathrm{AES}_k(n)$ is secure if δ is small, i.e., if AES_k meets its design goals.

In short, this paper guarantees that $m_n(r) + \mathrm{AES}_k(n)$ is as secure as AES up to 2^{64} messages. The best previous results did not handle nearly as many messages.

The importance of injectivity

Suppose that, in the above discussion of AES, I modify the definition of f by omitting the word "injective." Does the rest of the argument lead to the same security guarantee? No!

It is still true that the attacker's success chance against $m_n(r) + \mathrm{AES}_k(n)$ is bounded by the sum of two chances: first, the attacker's chance of distinguishing AES_k from f; second, the attacker's success chance against $m_n(r) + f(n)$. It is still true—and easy to prove, without the new techniques in this paper—that the second chance is small. But it is *not* true that AES was designed to make the first chance small. In fact, for $C = 2^{64}$, the first chance is *not* small. The attacker has a good chance of distinguishing AES_k from f by trying 2^{64} inputs and checking for collisions.

The importance of injectivity in this context was highlighted by Shoup in [19–Section 1] nearly ten years ago. As C and D grow, the usual theorems say "*nothing at all* about the security of the message authentication scheme," Shoup wrote, pointing out examples of this problem in the literature.

Unfortunately, the literature has continued to sprout problems of this type. Example: [9–Section 6.1] claims, for one message-authentication code, that any attack with success probability larger than "about 2^{-60}" has been "rigorously proven" to imply an attack that distinguishes AES from "a [uniform] family of random permutations." In fact, the security analysis considered the uniform distribution on all functions, not the uniform distribution on permutations; see [16–page 15]. The error is below 2^{-60} if $C + D < 2^{34}$, but neither [16] nor [9] puts any such limit on $C + D$. Apparently the security "guarantee" in [9] has not been proven.

Generalization

This paper considers much more general message-authentication codes of the form $(n, m) \mapsto h(m) + f(n)$. The main theorem of this paper, Theorem 5.1, is that $h(m) + f(n)$ is secure if (1) differential probabilities for h are small and (2) interpolation probabilities for f are small.

The "differential probabilities for h" are probabilities of the form $\Pr[h(m') = h(m) + g]$. The "interpolation probabilities for f" are probabilities of the form $\Pr[(f(n_1), \ldots, f(n_k)) = (x_1, \ldots, x_k)]$. See Appendix A for further discussion of terminology.

In particular, assume that f is a uniform random injective function from $\{n\}$ to a finite commutative group G, and that the differential probabilities for h are small. Then $h(m) + f(n)$ is secure against all attacks that see at most $\sqrt{\#G}$ authenticated messages. As a special case, if G is the set of 16-byte strings with a

group structure, then $h(m) + f(n)$ is secure against all attacks that see at most 2^{64} authenticated messages. Consequently $h(m) + \mathrm{AES}_k(n)$ is secure against any attacker who cannot break AES and who sees at most 2^{64} authenticated messages.

The form $h(m) \oplus f(n)$ for an authenticator, where f is a uniform random function, was introduced by Wegman and Carter in [22–Section 4]. Here $\oplus$ is vector addition modulo 2. Brassard in [10] considered $h(m) \oplus f(n)$ where f is a random function determined by a short key. Shoup in [19], as discussed above, considered $h(m) \oplus f(n)$ where f is a uniform random injective function.

There are many choices of h in the literature. The choices for any particular output size (say 128 bits) vary in speed, entropy (e.g., 128 bits), maximum differential probability ϵ (e.g., $L/2^{128}$), et al. For example, Gilbert, MacWilliams, and Sloane in [12] proposed $m_1, m_2, \ldots, m_L \mapsto m_1 r_1 + m_2 r_2 + \cdots + m_L r_L$, which has $\epsilon = 1/2^{128}$ but a very long $r = (r_1, r_2, \ldots, r_L)$. This is, at first glance, just as fast as univariate polynomial evaluation, but in practice the large r creates a huge speed penalty: cache misses become much more common and much more expensive. Evaluating a polynomial in a few variables over a larger field achieves the same ϵ with a much smaller speed penalty. The larger field means a larger output size, but Bierbrauer, Johansson, Kabatianskii, and Smeets in [7–page 336] pointed out that the result could be safely truncated after an appropriate twist; the bandwidth savings of truncation may justify the computation cost of the twist. There is much more to say about the choice of h; see [4–Section 10] for a survey.

The more general shape $h(m)+f(n)$ for an authenticator, where $+$ can be any commutative group operation, accommodates choices of h that rely on addition in large characteristic rather than characteristic 2—in particular, functions that can take advantage of the high-speed multiplication circuits included in common CPUs. Several examples of such functions are "MMH," "hash127," "UMAC," "CWC-HASH," and my new "Poly1305"; see [13], [4], [16], [15], and [5].

Shoup stated his theorems only for $\oplus$, but his proof technique can be used in the same level of generality as mine. His proof technique breaks down as C passes $\sqrt{1/\epsilon}$, where ϵ is the maximum differential probability of h, whereas mine continues working until $\sqrt{\#G}$, where G is the authenticator group. The magnitude of the improvement depends on how far ϵ is from $1/\#G$. In the Gilbert-MacWilliams-Sloane system, for example, $\epsilon = 1/\#G$ and there is no improvement; in Poly1305, $\epsilon \approx 2^{25} L/\#G$ and there is a large improvement.

All of the security proofs in the literature rely on two-sided bounds for the interpolation probabilities for f. One computes lower bounds on the probability of any particular sequence of authenticators; one computes nearby upper bounds on the probability of that sequence of authenticators given h; one deduces that the authenticators reveal very little information about h, and hence very little information about the authenticator for a new message. See, e.g., [22–Section 4, Theorem] and [19–Appendix A, Lemma 1]. The heart of the improvement in this paper is a new "one-sided" proof strategy that moves directly from upper bounds for f and h to upper bounds on the attacker's chance of success.

2 Protocol

This section describes a very general message-authentication protocol. Section 3 formalizes the notion of an attack on the protocol. Section 5 analyzes the success chance of all attacks.

The protocol has several parameters:

- G, a finite commutative group of **authenticators**. I will always write the group operation as $+$. (More general groups, or even loops, would suffice, but I see no application of the extra generality.) Typical example: G is the set of 16-byte strings, with the group operation being exclusive-or. Another example: G is the set $\{0, 1, 2, \ldots, 2^{128} - 1\}$, with the group operation being addition modulo 2^{128}.
- M, a nonempty set of **messages**. Typical example: M is the set of all strings of bytes. Another example: M is the set of all strings of at most 1024 bytes.
- N, a finite set of **nonces**, with $\#N \leq \#G$. Typical example: N is the set $\{1, 2, 3, \ldots, 2^{32} - 1\}$. Another example: N is the set of 16-byte strings.

The protocol has several participants:

- A **message generator** creates messages.
- A **nonce generator** accepts messages m from the message generator and attaches a nonce n to each message m. The nonce generator must never use the same nonce for two different messages: if it generates (n_1, m_1) and (n_2, m_2), and if $m_1 \neq m_2$, then n_1 must not equal n_2. This uniqueness rule is automatically satisfied if the nonce generator uses nonce 1 for the first message, nonce 2 for the second message, etc.
- A **sender** accepts pairs (n, m) from the nonce generator and attaches an authenticator a to each pair, as discussed below.
- A **network** accepts a sequence of vectors (n, m, a) from the sender and transmits a sequence of vectors (n', m', a'). Perhaps the sequence of vectors transmitted is the same as the sequence of vectors sent; perhaps not.
- A **receiver** receives vectors (n', m', a') from the network. It accepts (n', m') if a' is the authenticator that the sender would have attached to (n', m'); otherwise it discards (n', m').

If the network transmits exactly what the sender sent, then the pairs (n, m) accepted by the receiver are exactly the pairs (n, m) given to the sender; but what if the network makes changes? The objective of the protocol is **forgery elimination**: ensuring that each pair (n', m') accepted by the receiver is one of the pairs (n, m) that was authenticated by the sender.

Users also want additional protocol features:

- The receiver should notice if the network repeats messages or transmits messages out of order. One standard way to do this is for the nonce generator to use increasing nonces (in some specified ordering of the set N), and for the receiver to discard (n', m', a') unless n' is larger than the last accepted nonce.

- The receiver should notice if the network loses a message. There's no way to recover if the network is losing all messages, but there are retransmission protocols that eventually succeed in transmitting all data if the network delivers (e.g.) 1% of all messages.

But this paper focuses on the cryptographic problem of forgery elimination.

The sender's authenticator for a pair (n, m) is $h(m) + f(n)$: i.e., the sender gives $(n, m, h(m) + f(n))$ to the network. Here h is a random function from M to G, and f is a random function from N to G. The pair (f, h) is a secret shared by the sender and receiver; this means that the actions of the message generator, nonce generator, and network are independent of (f, h). In particular, if the message generator encrypts messages, it does so using a key independent of (f, h). The proof strategy in this paper can be extended to cover protocols that reuse f for encryption, as long as separate f inputs are used for encryption and for authentication; but that extension is not included in the statement of Theorem 5.1.

3 Attacks

The combined behavior of the message generator, nonce generator, and network is called an "attack." The attack creates messages; it creates nonces, subject to the rule that nonces never repeat; it inspects the authenticators provided by the sender; and it provides some number of forgery attempts to the receiver. The network is presumed to be able to provide data to the message generator and nonce generator, so each message can depend on previous authenticators. The attacker is presumed to be able to tell whether the receiver has accepted a forgery attempt.

More formally: An **attack** is an algorithm given oracle access to two functions S and R. The algorithm feeds **chosen messages** to the first oracle:

- The algorithm chooses a nonce n_1 and message m_1. The algorithm issues the query (n_1, m_1) and receives an authenticator $a_1 = S(n_1, m_1)$.
- The algorithm then chooses a nonce n_2 and message m_2, obeying the rule that $n_2 \neq n_1$ if $m_2 \neq m_1$. The algorithm issues the query (n_2, m_2) and receives an authenticator $a_2 = S(n_2, m_2)$.
- The algorithm then chooses a nonce n_3 and message m_3, obeying the rule that $n_3 \neq n_1$ if $m_3 \neq m_1$, and the rule that $n_3 \neq n_2$ if $m_3 \neq m_2$. The algorithm issues the query (n_3, m_3) and receives an authenticator $a_3 = S(n_3, m_3)$.
- The algorithm continues in this way for any number of messages.

Meanwhile, the algorithm feeds any number of **forgery attempts** (n', m', a') to the second oracle, receiving responses $R(n', m', a')$.

The attack **succeeds against** S **and** R if at least one forgery attempt (n', m', a') has $R(n', m', a') = 1$ with (n', m') different from the previous queries $(n_1, m_1), (n_2, m_2), \ldots$ to the first oracle.

Attacks against this system

Take, in particular, $S(n,m) = h(m) + f(n)$ and $R(n,m,a) = [a = h(m) + f(n)]$; i.e., $R(n,m,a) = 1$ if $a = h(m) + f(n)$ and $R(n,m,a) = 0$ if $a \neq h(m) + f(n)$. Is there an attack that succeeds against S and R with noticeable probability?

Theorem 5.1 states, under certain assumptions on f and h, that the answer is no. The receiver is overwhelmingly likely to discard every forgery—no matter how the message generator chooses messages; no matter how the nonce generator chooses unique nonces; no matter how the network chooses forgeries.

The rest of this section discusses the strength of this theorem, under the same assumptions on f and h.

Forgeries versus selective forgeries

A selective forgery is a forged message chosen *in advance* by the attacker. Some protocols prevent selective forgeries but allow attackers to find authenticators for random-looking messages. These protocols assume—often incorrectly—that random-looking messages will not cause any damage. In contrast, $h(m) + f(n)$ rejects *all* forgeries.

Attacks versus blind attacks

Some protocols prevent blind attacks but allow forgeries when attackers can inspect authenticated messages. (Trivial example: use a secret password as an authenticator for every message.) In contrast, $h(m) + f(n)$ rejects all forgeries even after the attacker sees a large number of authenticated messages.

Chosen messages versus known messages

Some protocols are secure for some message generators but insecure for others. An attacker who can influence the message generator can often obtain enough information to forge messages. In contrast, $h(m) + f(n)$ rejects all forgeries no matter what the message generator does.

Of course, if an attacker can convince the message generator to produce a message, then he does not need to forge an authenticator for that message. An easily corrupted message generator is often a problem. It is, however, not the cryptographic problem considered in this paper.

Receiver interaction

As pointed out by Bellare, Goldreich, and Mityagin in [3], some (admittedly unrealistic) protocols are secure against an attacker carrying out a single forgery attempt but insecure against an attacker that tries several forgery attempts. In contrast, the security bound for $h(m) + f(n)$ is linear in the number of forgery attempts.

The crucial point, as emphasized by Bellare et al., is that the attacker can recognize all (n', m', a') that will be accepted by the receiver without being forgeries: namely, the results $(n_1, m_1, a_1), (n_2, m_2, a_2), \ldots$ already obtained from the sender. In other words, the only way an attacker can learn anything new from the receiver is by succeeding at a forgery. Thus receiver interaction does not

improve the attacker's success probability. Receiver interaction can change the *number* of successful forgeries if the attacker succeeds, but this paper guarantees that the attacker will not succeed in the first place.

4 Interpolation Probabilities

Let f be a random function from N to G. The hypothesis on f in Section 5 is that f has maximum k-interpolation probability on the scale of $1/\#G^k$, for various $k \in \{0, 1, \ldots, \#N\}$. Here the **maximum k-interpolation probability of f** is the maximum, for all $x_1, x_2, \ldots, x_k \in G$ and all distinct $n_1, n_2, \ldots, n_k \in N$, of the probability that $(f(n_1), f(n_2), \ldots, f(n_k)) = (x_1, x_2, \ldots, x_k)$.

This section proves that this condition is satisfied by a uniform random function and by a uniform random injective function.

Theorem 4.1. *Let f be a uniform random function from a finite set N to a finite set G. Assume that $\#N \leq \#G$. Then f has maximum k-interpolation probability $1/\#G^k$ for each $k \in \{0, 1, \ldots, \#N\}$.*

Proof. $(f(n_1), f(n_2), \ldots, f(n_k)) = (x_1, x_2, \ldots, x_k)$ with probability $1/\#G^k$. □

Theorem 4.2. *Let f be a uniform random injective function from a finite set N to a finite set G. Assume that $\#N \leq \#G$. Then f has maximum k-interpolation probability at most $(1 - (k-1)/\#G)^{-k/2}/\#G^k$ for each $k \in \{0, 1, \ldots, \#N\}$.*

Proof. Fix distinct $n_1, n_2, \ldots, n_k \in N$. Fix $x_1, x_2, \ldots, x_k \in G$.

Case 1: There are collisions in $x_1, x_2, \ldots, x_k$. Then $(f(n_1), \ldots, f(n_k)) = (x_1, \ldots, x_k)$ with probability 0.

Case 2: There are no collisions. Then $f(n_1) = x_1$ with probability $1/\#G$; if that happens then $f(n_2) = x_2$ with conditional probability $1/(\#G - 1)$; if that happens then $f(n_3) = x_3$ with conditional probability $1/(\#G - 2)$; and so on. The probability that $(f(n_1), f(n_2), \ldots, f(n_k)) = (x_1, x_2, \ldots, x_k)$ is exactly $\prod_{0 \leq i \leq k-1} 1/(\#G - i) = \sqrt{\prod_{0 \leq i \leq k-1} 1/(\#G - i)(\#G - (k-1-i))} \leq \sqrt{\prod_{0 \leq i \leq k-1} 1/(\#G)^2(1 - (k-1)/\#G)} = \sqrt{(1 - (k-1)/\#G)^{-k}/(\#G)^{2k}}$. □

5 The Main Theorem

Theorem 5.1 is the main theorem of this paper: $(n, m) \mapsto h(m) + f(n)$ is secure if h has small differential probabilities and f has small interpolation probabilities.

Theorems 5.2 and 5.3 consider two special cases: a uniform random function f, and a uniform random injective function f.

Theorem 5.4 proves that $(n, m) \mapsto h(m) + \mathrm{AES}_k(n)$ is secure if h has small differential probabilities and AES_k is secure, i.e., AES_k is difficult to distinguish from a uniform random injective function.

Theorem 5.1. *Let h be a random function from a nonempty set M to a finite commutative group G. Let f be a random function from a finite set N to G. Let C and D be positive integers. Assume that $C+1 \leq \#N \leq \#G$. Assume, for all $g \in G$ and all distinct $m, m' \in M$, that $h(m) = h(m')+g$ with probability at most ϵ. Assume that f has maximum C-interpolation probability at most $\delta/\#G^C$ and maximum $(C+1)$-interpolation probability at most $\delta\epsilon/\#G^C$. Assume that h and f are independent. Then any attack using at most C distinct chosen messages and at most D forgery attempts succeeds against $(n, m) \mapsto h(m) + f(n)$ and $(n, m, a) \mapsto [a = h(m) + f(n)]$ with probability at most $D\delta\epsilon$.*

Proof. Standard reduction #1: It suffices to consider $D = 1$, for the following reason. For $D > 1$, split the attack into two pieces:

- The first piece is the original attack with one change: it stops immediately after the first forgery attempt (if there are any forgery attempts). This piece uses at most C distinct chosen messages and at most 1 forgery attempt.
- The second piece is the original attack with one change: it simulates the the first forgery attempt internally (if there are any forgery attempts) rather than sending the forgery attempt as an oracle query. The simulator returns 1 if the forgery attempt n', m', a' matches an authenticator a' already provided in response to a chosen message m' with nonce n'; otherwise the simulator returns 0. This piece uses at most C distinct chosen messages and at most $D-1$ forgery attempts.

Success of the original attack on its first forgery attempt is equivalent to success of the first piece—which occurs with probability at most $\delta\epsilon$. Failure of the original attack on its first forgery attempt, but success on a subsequent attempt, implies success of the second piece—which, by induction on D, occurs with probability at most $(D-1)\delta\epsilon$. Therefore the original attack succeeds with probability at most $D\delta\epsilon$.

Standard reduction #2: If there are no forgery attempts then the attack succeeds with probability $0 \leq \delta\epsilon$. Assume from now on that there is exactly one forgery attempt.

Standard reduction #3: If the attack chooses any messages after the forgery attempt, modify it to discard those oracle queries; this has no effect on the attack's success chance. Assume from now on that all chosen messages are issued before the forgery attempt.

Standard reduction #4: If the attack might use fewer than C distinct chosen messages, modify it to use additional chosen messages with new nonces and to discard the results; new nonces are available since $\#N \geq C$, and at least one message is available since $\#M \geq 1$. Assume from now on that the attack uses exactly C distinct chosen messages.

Standard reduction #5: If the attack might repeat chosen messages, modify it to cache queries and responses to the sender oracle. Assume from now on that the attack does not repeat chosen messages.

Write (n_i, m_i) for the ith query to the sender oracle. Then $n_1, n_2, \ldots, n_C$ are distinct. Write a_i for the ith response from the oracle, when the attack is

applied to $(n, m) \mapsto h(m) + f(n)$; then $a_i = h(m_i) + f(n_i)$. Write (n', m', a') for the attempted forgery.

Everything that the attack does is determined by (1) an infinite sequence b of coin flips, by definition independent of h and f, and (2) the sequence of sender responses $a_1, a_2, \ldots, a_C$. In particular, $n_1, n_2, \ldots, n_C, m_1, m_2, \ldots, m_C, n', m', a'$ are equal to various functions evaluated at $b, a_1, a_2, \ldots, a_C$. Furthermore, $f(n_i)$ is determined by a_i and $h(m_i)$, so $f(n_i)$ is equal to a function evaluated at $h, b, a_1, a_2, \ldots, a_C$.

Fix $(g_1, g_2, \ldots, g_C) \in G^C$. Define X as the following event: (n', m', a') is a successful forgery and $(a_1, a_2, \ldots, a_C) = (g_1, g_2, \ldots, g_C)$. It suffices to show that event X has probability at most $\delta\epsilon/\#G^C$.

(A referee suggests some added emphasis: I am considering the probability that the forgery attempt succeeds *and* the authenticators match $(g_1, g_2, \ldots, g_C)$. Previous proofs considered the probability that the forgery attempt succeeds *given that* the authenticators match $(g_1, g_2, \ldots, g_C)$.)

Define p as the probability that b satisfies the following measurable constraint: if $(a_1, a_2, \ldots, a_C) = (g_1, g_2, \ldots, g_C)$ then $n' \notin \{n_1, n_2, \ldots, n_C\}$. I claim, for each b satisfying the constraint and for each h, that f has conditional probability at most $\delta\epsilon/\#G^C$ of producing event X.

Indeed, assume that b satisfies the constraint, that (n', m', a') is a successful forgery, and that $(a_1, a_2, \ldots, a_C) = (g_1, g_2, \ldots, g_C)$. Then $\#\{n_1, \ldots, n_C, n'\} = C + 1$, and the pairs $(n_1, f(n_1)), \ldots, (n_C, f(n_C)), (n', f(n'))$ are equal to various functions evaluated at $h, b, g_1, g_2, \ldots, g_C$. By hypothesis, f is independent of h; f is also independent of b; and $g_1, g_2, \ldots, g_C$ are fixed. The conditional probability of f interpolating those pairs is at most the maximum $(C + 1)$-interpolation probability of f, which by hypothesis is at most $\delta\epsilon/\#G^C$.

I also claim, for each b *not* satisfying the constraint, that h has conditional probability at most ϵ of satisfying a necessary differential condition; and, for each b and each qualifying h, that f has conditional probability at most $\delta/\#G^C$ of producing event X.

Indeed, assume that b does not satisfy the constraint, that $(a_1, a_2, \ldots, a_C) = (g_1, g_2, \ldots, g_C)$, and that (n', m', a') is a successful forgery. Then $n' = n_i$ for a unique i; note that $m' \neq m_i$. Next $a' = h(m') + f(n_i)$ and $a_i = h(m_i) + f(n_i)$ so $h(m_i) - h(m') = a_i - a'$. The inputs m_i, m' and the output $a_i - a'$ are equal to various functions evaluated at $b, g_1, g_2, \ldots, g_C$, and thus are independent of h; by hypothesis, h satisfies the condition $h(m_i) - h(m') = a_i - a'$ with probability at most ϵ. Furthermore, the pairs $(n_1, f(n_1)), (n_2, f(n_2)), \ldots, (n_C, f(n_C))$ are equal to various functions evaluated at $h, b, g_1, g_2, \ldots, g_C$; f is once again independent of $h, b, g_1, g_2, \ldots, g_C$; so the conditional probability of f interpolating those pairs is at most the maximum C-interpolation probability of f, which by hypothesis is at most $\delta/\#G^C$.

The total probability of event X is at most $p(\delta\epsilon/\#G^C) + (1-p)(\epsilon)(\delta/\#G^C) = \delta\epsilon/\#G^C$. □

Theorem 5.2. *Let h be a random function from a nonempty set M to a finite commutative group G. Let f be a uniform random function from a finite set*

N to G. Let C and D be positive integers. Assume that $C+1 \leq \#N \leq \#G$. Assume, for all $g \in G$ and all distinct $m, m' \in M$, that $h(m) = h(m') + g$ with probability at most ϵ. Assume that h and f are independent. Assume that $\epsilon \geq 1/\#G$. Then any attack using at most C distinct chosen messages and at most D forgery attempts succeeds against $(n, m) \mapsto h(m) + f(n)$ and $(n, m, a) \mapsto [a = h(m) + f(n)]$ with probability at most $D\epsilon$.

The condition $\epsilon \geq 1/\#G$ is redundant if $\#M \geq 2$: any distinct $m, m' \in M$ have $\#G$ possibilities for $h(m) - h(m')$, each occurring with probability at most ϵ by hypothesis, so $1 \leq \epsilon \#G$.

Proof. Write $\delta = 1$. Then f has maximum C-interpolation probability $1/\#G^C = \delta/\#G^C$, and maximum $(C+1)$-interpolation probability $1/\#G^{C+1} \leq \delta\epsilon/\#G^C$, by Theorem 4.1. By Theorem 5.1, the attack succeeds with probability at most $D\delta\epsilon = D\epsilon$. □

Theorem 5.3. *Let h be a random function from a nonempty set M to a finite commutative group G. Let f be a uniform random injective function from a finite set N to G. Let C and D be positive integers. Assume that $C+1 \leq \#N \leq \#G$. Assume, for all $g \in G$ and all distinct $m, m' \in M$, that $h(m) = h(m') + g$ with probability at most ϵ. Assume that h and f are independent. Assume that $\epsilon \geq 1/\#G$. Then any attack using at most C distinct chosen messages and at most D forgery attempts succeeds against $(n, m) \mapsto h(m) + f(n)$ and $(n, m, a) \mapsto [a = h(m) + f(n)]$ with probability at most $D(1 - C/\#G)^{-(C+1)/2}\epsilon$.*

In the special case $C = \lfloor \sqrt{\#G} \rfloor$, the extra factor $(1-C/\#G)^{-(C+1)/2}$ is below 1.7 for all reasonably large G; it converges to $\exp(1/2) \approx 1.64872$ as $\#G \to \infty$.

Proof. Write $\delta = (1 - C/\#G)^{-(C+1)/2}$. By Theorem 4.2, f has maximum C-interpolation probability at most $(1 - (C-1)/\#G)^{-C/2}/\#G^C \leq \delta/\#G^C$. By Theorem 4.2 again, f has maximum $(C+1)$-interpolation probability at most $(1 - C/\#G)^{-(C+1)/2}/\#G^{C+1} \leq \delta\epsilon/\#G^C$. By Theorem 5.1, the attack succeeds with probability at most $D\delta\epsilon$. □

Theorem 5.4. *Let G be the set of 16-byte strings with a group structure. Let k be a random AES key. Let h be a random function from a nonempty set M to G. Assume that the distribution of h is computable. Let C and D be positive integers. Assume that $C + 1 \leq 2^{128}$. Assume, for all $g \in G$ and all distinct $m, m' \in M$, that $h(m) = h(m') + g$ with probability at most ϵ. Assume that h and k are independent. Assume that $\epsilon \geq 1/2^{128}$. Let A be an attack using at most C distinct chosen messages and at most D forgery attempts. Assume that A succeeds against $(n, m) \mapsto h(m) + \mathrm{AES}_k(n)$ and $(n, m, a) \mapsto [a = h(m) + \mathrm{AES}_k(n)]$ with probability γ. Define A' as the algorithm that, given an oracle for a function f, chooses h randomly, applies A to $(n, m) \mapsto h(m) + f(n)$ and $(n, m, a) \mapsto [a = h(m) + f(n)]$, and prints 1 if A succeeded. Then A' distinguishes AES_k from a uniform random permutation of G with probability at least $\gamma - D(1 - C/2^{128})^{-(C+1)/2}\epsilon$, using at most $C + D$ oracle queries.*

Consequently, A succeeds against $(n, m) \mapsto h(m) + \mathrm{AES}_k(n)$ and $(n, m, a) \mapsto [a = h(m) + \mathrm{AES}_k(n)]$ with probability at most $\delta + D(1 - C/2^{128})^{-(C+1)/2}\epsilon$, where δ is the probability that an algorithm as fast as A' can distinguish AES_k from a uniform random permutation of G.

Proof. A' makes one oracle query for each chosen message from A, and one oracle query for each attempted forgery from A, for a total of at most $C + D$ oracle queries.

When A' is given an oracle for AES_k, it applies A to $(n, m) \mapsto h(m)+\mathrm{AES}_k(n)$ and $(n, m, a) \mapsto [a = h(m) + \mathrm{AES}_k(n)]$, so it prints 1 with probability γ by hypothesis.

When A' is given an oracle for a uniform random permutation f of G, it applies A to $(n, m) \mapsto h(m) + f(n)$ and $(n, m, a) \mapsto [a = h(m) + f(n)]$, so it prints 1 with probability at most $D(1 - C/2^{128})^{-(C+1)/2}\epsilon$ by Theorem 5.3.

Therefore A' distinguishes AES_k from a uniform random permutation of G with probability at least $\gamma - D(1 - C/2^{128})^{-(C+1)/2}\epsilon$. □

References

1. —, *20th annual symposium on foundations of computer science*, IEEE Computer Society, New York, 1979. MR 82a:68004.
2. —, *Announcing request for candidate algorithm nominations for the Advanced Encryption Standard (AES)* (1997). URL: `http://csrc.nist.gov/CryptoToolkit/aes/pre-round1/aes_9709.htm`.
3. Mihir Bellare, Oded Goldreich, Anton Mityagin, *The power of verification queries in message authentication and authenticated encryption* (2004). URL: `http://eprint.iacr.org/2004/309`.
4. Daniel J. Bernstein, *Floating-point arithmetic and message authentication*, to be incorporated into author's *High-speed cryptography* book. URL: `http://cr.yp.to/papers.html#hash127`. ID `dabadd3095644704c5cbe9690ea3738e`.
5. Daniel J. Bernstein, *The Poly1305-AES message-authentication code* (2005); Proceedings of Fast Software Encryption 2005, to appear. URL: `http://cr.yp.to/papers.html#poly1305`. ID `0018d9551b5546d97c340e0dd8cb5750`.
6. Daniel J. Bernstein, *A short proof of the unpredictability of cipher block chaining* (2005). URL: `http://cr.yp.to/papers.html#easycbc`. ID `24120a1f8b92722b5e15fbb6a86521a0`.
7. Jürgen Bierbrauer, Thomas Johansson, Gregory Kabatianskii, Ben Smeets, *On families of hash functions via geometric codes and concatenation*, in [20] (1994), 331–342. URL: `http://cr.yp.to/bib/entries.html#1994/bierbrauer`.
8. Eli Biham (editor), *Fast Software Encryption '97*, Lecture Notes in Computer Science, 1267, Springer-Verlag, Berlin, 1997. ISBN 3–540–63247–6.
9. John Black, Shai Halevi, Alejandro Hevia, Hugo Krawczyk, Ted Krovetz, Phillip Rogaway, *UMAC: message authentication code using universal hashing* (2004). URL: `http://www.cs.ucdavis.edu/~rogaway/umac/index.html`.
10. Gilles Brassard, *On computationally secure authentication tags requiring short secret shared keys*, in [11] (1983), 79–86. URL: `http://cr.yp.to/bib/entries.html#1983/brassard`.

11. David Chaum, Ronald L. Rivest, Alan T. Sherman (editors), *Advances in cryptology: proceedings of Crypto 82*, Plenum Press, New York, 1983. ISBN 0–306–41366–3. MR 84j:94004.
12. Edgar N. Gilbert, F. Jessie MacWilliams, Neil J. A. Sloane, *Codes which detect deception*, Bell System Technical Journal **53** (1974), 405–424. ISSN 0005–8580. MR 55:5306. URL: `http://cr.yp.to/bib/entries.html#1974/gilbert`.
13. Shai Halevi, Hugo Krawczyk, *MMH: software message authentication in the Gbit/second rates*, in [8] (1997), 172–189. URL: `http://www.research.ibm.com/people/s/shaih/pubs/mmh.html`.
14. Neal Koblitz (editor), *Advances in cryptology—CRYPTO '96*, Lecture Notes in Computer Science, 1109, Springer-Verlag, Berlin, 1996.
15. Tadayoshi Kohno, John Viega, Doug Whiting, *CWC: a high-performance conventional authenticated encryption mode* (2004). URL: `http://www.cs.ucsd.edu/users/tkohno/papers/CWC/`.
16. Theodore Krovetz, *Software-optimized universal hashing and message authentication*, Ph.D. thesis, University of California at Davis, 2000. URL: `http://www.cs.ucdavis.edu/~rogaway/umac/`.
17. James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, Edward Roback, *Report on the development of the Advanced Encryption Standard (AES)*, Journal of Research of the National Institute of Standards and Technology **106** (2001). URL: `http://nvl.nist.gov/pub/nistpubs/jres/106/3/cnt106-3.htm`.
18. Victor Shoup, *On fast and provably secure message authentication based on universal hashing*, in [14] (1996), 313–328; see also newer version [19].
19. Victor Shoup, *On fast and provably secure message authentication based on universal hashing* (1996); see also older version [18]. URL: `http://www.shoup.net/papers`.
20. Douglas R. Stinson (editor), *Advances in cryptology—CRYPTO '93: 13th annual international cryptology conference, Santa Barbara, California, USA, August 22–26, 1993, proceedings*, Lecture Notes in Computer Science, 773, Springer-Verlag, Berlin, 1994. ISBN 3–540–57766–1, 0–387–57766–1. MR 95b:94002.
21. Mark N. Wegman, J. Lawrence Carter, *New classes and applications of hash functions*, in [1] (1979), 175–182; see also newer version [22]. URL: `http://cr.yp.to/bib/entries.html#1979/wegman`.
22. Mark N. Wegman, J. Lawrence Carter, *New hash functions and their use in authentication and set equality*, Journal of Computer and System Sciences **22** (1981), 265–279; see also older version [21]. ISSN 0022–0000. MR 82i:68017. URL: `http://cr.yp.to/bib/entries.html#1981/wegman`.

A Appendix: General Terminology

This section discusses various conflicts between (1) the terminology used in some cryptographic papers and (2) the standard terminology of probability theory used by a much larger community of mathematicians. This section also discusses my choice of terminology for a few concepts that are more specialized.

"Random" versus "uniform random"

A random element v of a finite set S is **uniform** if $\Pr[v = s] = 1/\#S$ for each $s \in S$. This terminology is standard in probability theory and is used in this paper.

Examples: A coin flip c—a random bit—is uniform if $\Pr[c = 0] = 1/2$ and $\Pr[c = 1] = 1/2$. If k is a uniform random 16-byte string then $(k, 0)$ is a non-uniform random 17-byte string; $k[0]$, the first byte of k, is a uniform random byte; AES_k is a non-uniform random permutation of the set of 16-byte strings.

Some cryptographic papers use the word "random" to mean uniform random. If these papers state theorems regarding a "random element of S," for example, then those theorems don't apply to a random element of $\{0, 1, 2, \ldots, 2^{127} - 2\}$ that's 0 twice as often as anything else—even though this slightly non-uniform distribution is much more widely used than the uniform distribution. If these papers state theorems regarding a "random RSA key" then those theorems are incompatible with every prime-generation algorithm that's actually used.

Some people try to work around this terminological deficiency by viewing all random variables as images of uniform random variables. My random element of $\{0, 1, 2, \ldots, 2^{127} - 2\}$ might be viewed as the reduction modulo $2^{127} - 1$ of a uniform random element of $\{0, 1, 2, \ldots, 2^{256} - 1\}$; maybe this is how it was generated in the first place.

The big problem with this workaround is that it buries every random variable in a pointless thicket of notation—one has to introduce an irrelevant input set and an irrelevant function instead of simply focusing on the resulting variable. For example, rather than simply stating a theorem for a random function h from messages to 16-byte strings, I'd have to state a theorem for a (uniform) random element r of some set R together with some function H that, for each r, gives me a function from messages to 16-byte strings.

"Random" versus "discrete random"

The set of values of a random variable is *not* required to be finite, or even countable. A **discrete** random variable is a random element of a countable set. This terminology is standard in probability theory and is used in this paper.

Consider, for example, the coin flips provided as an auxiliary input to a probabilistic algorithm: an infinite random sequence of bits $b = (b_1, b_2, b_3, \ldots)$. This random sequence is a non-discrete random variable; it has uncountably many values.

As a concrete example, consider the usual probabilistic algorithm to generate a uniform random element of $\{1, 2, 3\}$: flip two coins; if the results are $(0, 1)$ or $(1, 0)$ or $(1, 1)$, print 1 or 2 or 3 correspondingly and stop; otherwise try again. It is easy to prove that this algorithm succeeds with probability 1 (i.e., that $b = (0, 0, 0, 0, \ldots)$ with probability 0), that the algorithm flips 8/3 coins on average, etc. These are statements about non-discrete probabilities.

Some cryptographic papers use the word "random" to mean discrete random, thus excluding such fundamental objects as coin-flip sequences. This restriction is intolerable for any serious discussion of probabilistic algorithms.

Warning to undergraduates: $\Pr[b \in S]$, the probability that b is in S, is defined only for *some* sets S, namely the **measurable** sets. Here are the rules:

(1) the empty set is measurable;
(2) if S is measurable then its complement $\overline{S}$ is measurable;
(3) if $S_1, S_2, \ldots$ are measurable then $S_1 \cup S_2 \cup \cdots$ is measurable;
(4) if S is measurable then $\Pr[b \in S] \geq 0$;
(5) if S is measurable then $\Pr[b \in S] + \Pr[b \in \overline{S}] = 1$;
(6) $\Pr[b \in S_1 \cup S_2 \cup \cdots] = \Pr[b \in S_1] + \Pr[b \in S_2] + \cdots$ if $S_1, S_2, \ldots$ are disjoint measurable sets;
(7) if $u_1, \ldots, u_k$ are bits then $\{b : (b_1, \ldots, b_k) = (u_1, \ldots, u_k)\}$ is measurable and $\Pr[(b_1, \ldots, b_k) = (u_1, \ldots, u_k)] = 1/2^k$;
(8) nothing is measurable except as guaranteed above.

"Random" versus "independent random"

Random variables u, v, w are **independent** if the distribution of (u, v, w) is the product of the distribution of u, the distribution of v, and the distribution of w: i.e., $\Pr[(u, v, w) \in A \times B \times C] = \Pr[u \in A] \Pr[v \in B] \Pr[w \in C]$ for all measurable sets A, B, C. This terminology is standard in probability theory and is used in this paper.

For example, if k is a uniform random 16-byte string, then $k[0]$, $k[1]$, and $k[2]$ are independent uniform random bytes; $k[0]$, $k[1]$, and $k[0] \oplus k[1]$ are non-independent uniform random bytes; $k[0]$ and $k[0] \oplus k[1]$ are independent uniform random bytes; $(k[0], 0)$ and $(k[1], 0)$ are independent non-uniform random 2-byte strings.

Suppose Theorem X says "Let u and v be independent random bytes. Then u and v satisfy ...." I can apply Theorem X to the independent random bytes $k[1], k[2]$. I can apply it to the independent random bytes $k[0], k[0] \oplus k[1]$. I simply have to say "$k[0]$ and $k[0] \oplus k[1]$ are independent; therefore, by Theorem X, $k[0]$ and $k[0] \oplus k[1]$ satisfy ...."

Some cryptographic papers omit the word "independent" in many situations. For example, Theorem X would say "Let u and v be random bytes. Then u and v satisfy ..." implicitly also requiring that u and v be independent. The scope of this implicit independence is unclear to me: for example, if the proof begins "Note that if r is a random byte then ...," then is r implicitly required to be independent of u and v? The obvious way to avoid confusion is to make all independence assumptions explicit.

Distributions versus random variables

What is a random variable?

A random element v of X is, intuitively, a function to X from the set of possible universes. The value that v takes in a possible universe u is exactly the function value $v(u)$. For example, a coin flip is a function that assigns 0 to some possible universes and 1 to other possible universes.

To formalize this, we fix a probability space Pr—intuitively, the set of possible universes, although this intuition does not constrain the definition. A **random**

element of X, where X is a measurable space, is a measurable function from Pr to X. This terminology is standard in probability theory and is used in this paper.

(Notes for undergraduates: A **measurable space** is a set together with a designated collection of measurable subsets satisfying rules 1, 2, 3 above. A **probability space** is a set together with a designated collection of measurable subsets S and probabilities $\Pr[S]$ satisfying rules 1, 2, 3, 4, 5, 6 above. A function v is **measurable** if $\{u \in \Pr : v(u) \in S\}$ is measurable for every measurable set S.)

Random variables can be combined to produce new random variables. For example, if v is a random element of X and w is a random element of Y then (v, w), the function that takes u to $(v(u), w(u))$, is a random element of $X \times Y$. Similarly, if φ is a measurable function from X to Y, then $\varphi(v)$, the function that takes u to $\varphi(v(u))$, is a random element of Y.

Consider, for example, the measurable function $s \mapsto s[0]$ that extracts the first byte of a 16-byte string. This function induces a function $k \mapsto k[0]$ that extracts a *random* byte from a *random* 16-byte string: the composition of k (a function from Pr to 16-byte strings) with [0] (a function from 16-byte strings to bytes) is $k[0]$ (a function from Pr to bytes).

Some cryptographers forbid all use of random variables. For example, one is forbidden from considering a random function f and defining the maximum 2-interpolation probability of f as the maximum, over all x_1, x_2 and all distinct n_1, n_2, of $\Pr[(f(n_1), f(n_2)) = (x_1, x_2)]$. One is forced to use distributions instead: consider a distribution F on the set of functions, and define the maximum 2-interpolation probability of F as the maximum, over all x_1, x_2 and all distinct n_1, n_2, of the fraction of functions f in F such that $(f(n_1), f(n_2)) = (x_1, x_2)$.

The most glaring deficiency in this approach is its inability to discuss the dependence of separate random variables. The independence of p and q is not determined by the distribution of p and the distribution of q; in any situation where p and q might be dependent, these papers are forced to start from the distribution of the pair (p, q). How, then, does one feed p by itself to a lower-level theorem? One has to average the joint distribution to obtain the distribution of p, then apply the lower-level theorem, then undo the averaging; or generalize the lower-level theorem to allow a joint distribution as input—effectively reinventing the concept of random variables but with a much less pleasant notation.

Interpolation probabilities, collision probabilities, differential probabilities

The following concepts are much more specialized than the fundamental concepts of probability theory discussed earlier in this appendix. They are nevertheless sufficiently common that the community would benefit from settling on good terminology for them.

Let f be a random function from a set X to a finite set Y. Consider the probability that f interpolates the points $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$, where $x_1, x_2, \ldots, x_k$ are distinct: i.e., that $(f(x_1), f(x_2), \ldots, f(x_k)) = (y_1, y_2, \ldots, y_k)$.

This is what I call an **interpolation probability**, and more specifically a k-interpolation probability.

Now consider the sum of 2-interpolation probabilities along the diagonal: fix distinct $x_1, x_2 \in X$, and consider the probability that $f(x_1) = f(x_2)$. This is what I call a **collision probability**. More generally, assume that Y is a commutative group, fix $g \in Y$, and consider the probability that $f(x_1) - f(x_2) = g$. This is what I call a **differential probability**.

In the context of message authentication, it is useful to have upper bounds on interpolation probabilities and upper bounds on differential probabilities, as illustrated by this paper. Other authenticators can use upper bounds on collision probabilities. It is also useful to have *lower* bounds on interpolation probabilities, as illustrated by [6].

Many papers write "h is ϵ-almost-universal" (sometimes replacing h by its distribution) to mean that all collision probabilities of h are below ϵ. There are several flaws in this "ϵ-almost-universal" terminology:

- The phrase "almost universal" is highly non-descriptive. Readers who have not seen the definition cannot even begin to guess what it refers to. Readers who have seen the definition need to expend unnecessary mental energy to remember it.
- The phrase "almost universal" provides no way to refer to individual collision probabilities, lower bounds for collision probabilities, etc. The same papers often end up talking about "collision probabilities" anyway.
- The phrase "almost universal" begs the question of what "almost" refers to. The answer is that "h is universal" is reserved for the special case $\epsilon = 1/\#Y$, which is close to (although not exactly) the minimum achievable. This terminology misleads readers into believing that the special case is important; in fact, the general case is far more important.

Some papers on Wegman-Carter authenticators use the phrase "h is ϵ-almost-xor-universal" to mean that all differential probabilities of h are below ϵ, in the special case of the group operation being $\oplus$. Similar criticisms apply to this phrase. A few papers refer to the general case using the phrase "h is ϵ-almost-Δ-universal," which is a poor choice of terminology for yet another reason: the letter Δ is part of the terminology, not a modifiable variable name.

3-Move Undeniable Signature Scheme

Kaoru Kurosawa[1] and Swee-Huay Heng[2]

[1] Ibaraki University, 4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan
kurosawa@cis.ibaraki.ac.jp
[2] Multimedia University, Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia
shheng@mmu.edu.my

Abstract. In undeniable signature schemes, zero-knowledgeness and non-transferability have been identified so far. In this paper, by separating these two notions, we show the first 3-move confirmation and disavowal protocols for Chaum's undeniable signature scheme which is secure against active and concurrent attacks. Our main observation is that while the signer has one public key and one secret key, there exist two witnesses in the confirmation and disavowal proofs of Chaum's scheme.

Keywords: Undeniable signature, witness indistinguishability.

1 Introduction

1.1 Background and Motivation

The concept of undeniable signatures was due to Chaum and van Antwerpen [11]. As opposed to the standard digital signatures which are universally verifiable, the validity of undeniable signatures can be verified only with the signer's consent, by engaging interactively or non-interactively in either a confirmation protocol or a disavowal protocol. There have been a wide range of research covering a variety of different schemes for undeniable signatures over the past 15 years. Among others, we have [8, 3, 10, 9, 22, 15, 20, 24, 6, 19, 18, 23, 5, 25, 26]. Most of these schemes are discrete logarithm based, with the exception of a few RSA-based schemes [20, 19, 18], a pairing-based (identity-based) scheme [23] and some other schemes [5, 25, 26]. These schemes possess variable degrees of security and additional features such as convertibility [3, 15, 24, 18], designated verifier technique [22], designated confirmer technique [9, 27], and so on. At the same time, undeniable signatures also find various applications in cryptography such as in licensing softwares (this is in fact the original motivation of Chaum and van Antwerpen) [11], electronic cash [12, 4, 28], electronic voting and auctions.

An undeniable signature scheme is said to be secure (against active attack) if it is unforgeable, invisible and the confirmation and disavowal protocols are both zero-knowledge. The zero-knowledgeness is required to make undeniable signatures non-transferable, which is indeed the purpose of undeniable signature schemes. Further, it is believed that a 3-move protocol cannot be zero-knowledge from the result of [17].

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 181–197, 2005.

Therefore, no 3-move undeniable signature scheme which is secure against active attack is known. In fact, in the existing literature, the zero-knowledge confirmation protocol is at least 4 moves. No constant moves zero-knowledge disavowal protocol has been known so far.

This is mainly because zero-knowledgeness and non-transferability have been identified so far implicitly or explicitly. In other words, the search for a 3-move undeniable signature scheme which is provably secure against active attack remains as an challenging open problem since the introduction of the concept of undeniable signatures in 1989.

1.2 Our Contributions

We say that an undeniable signature scheme is "3-move" if the confirmation protocol and the disavowal protocol are both 3-move (where the signer S sends a to the verifier V, V sends b to S and S sends c to V).

In this paper, we propose the first "3-move" undeniable signature scheme which is provably secure against active and concurrent attacks, by exploiting the fact that DH-tuples possess two witnesses, and also that non DH-tuples possess two witnesses. It is achieved

- by separating two notions, zero-knowledgeness and non-transferability, and
- by incorporating the concept of witness indistinguishability [16] in a novel way.

A naive approach for witness indistinguishability would be to use two public keys, where the two corresponding secret keys are two witnesses, as suggested by Feige and Shamir in [16]. More precisely, the signer issues two undeniable signatures σ_1 and σ_2 on a message m. He then proves that σ_1 is valid or σ_2 is valid by a witness indistinguishable protocol. Unfortunately, this approach does not work. This is because, from De Morgan's law ($\overline{X \vee Y} = \overline{X} \wedge \overline{Y}$), both the confirmation protocol and the disavowal protocol cannot be witness indistinguishable simultaneously. For more details, see Section 6. Further, such a two public-key scheme would be costly.

On the other hand, we show 3-move confirmation and disavowal protocols for Chaum's undeniable signature scheme [8], where the signer has only one public key. Our main idea is as follows. In the confirmation (disavowal) protocol, the signer proves that a tuple (g, g^u, g^v, g^w) is a DH-tuple (non DH-tuple). Now observe that a DH-tuple (g, g^u, g^v, g^{uv}) has two witnesses, u and v. A similar observation holds for non DH-tuples too. Thus, our main observation is that while the signer has one public key and one secret key, there exist two witnesses in the confirmation and disavowal proofs of Chaum's scheme. This allows us to use the concept of witness indistinguishability (WI) in the confirmation and disavowal protocols. As a result, we manage to circumvent the problem encountered earlier in the naive approach.

More precisely, in order to prove that (g, g^u, g^v, g^{uv}) is a DH-tuple, knowledge of either one of the two witnesses, i.e. u or v is sufficient. This observation is critical in the simulation of the confirmation/disavowal oracle in the security analyses.

Chaum's original scheme (which does not employ a cryptographic hash function) is not secure as it succumbed to the basic multiplicative attacks. Therefore, we apply the above idea on the full-domain hash (FDH) [14] variant of Chaum's scheme so that we can treat the hash function as a random oracle in the security analyses. In this scheme, the signer has a single public key $y = g^x$ and a single secret key x. Remember that the signer does not have to prove that he knows x in the confirmation protocol. All he needs to prove is the validity of a signature σ on a message m under the public key y, i.e. he proves that a given tuple $(g, y, H(m), \sigma)$ is a DH-tuple where $\sigma = H(m)^x$ and H is a random oracle. We notice that the DH-tuple has two witnesses by accident in this case. The same argument applies in proving the invalidity of a signature $\sigma \neq H(m)^x$, i.e. by proving that $(g, y, H(m), \sigma)$ is a non DH-tuple using a disavowal protocol.

Traditionally, two main security notions for undeniable signatures are the notion of existential unforgeability and invisibility under adaptive chosen message attack. The existential unforgeability of our proposed scheme is equivalent to the computational Diffie-Hellman (CDH) problem while it is invisible assuming the hardness of the decisional Diffie-Hellman (DDH) problem.

In this paper, we also introduce another important security notion with regard to undeniable signatures, namely, the security against impersonation attack. As all of us are aware that the purpose of undeniable signature scheme is to construct a signature which is non-transferable. This is equivalent to prevent impersonation by employing confirmation and disavowal protocols. This security notion has not been formalized so far mainly because zero-knowledgeness implies non-transferability. Since our newly proposed scheme is a novel work separating zero-knowledgeness and non-transferability while adopting the witness-indistinguishability property, we are led to this particular security notion. We manage to prove that the security against impersonation attack of our proposed scheme is equivalent to the discrete logarithm (DLOG) problem.

A brief summary of security analyses of our newly proposed scheme is given in the following table. The table holds in the random oracle model while our confirmation (disavowal) protocol is a WI protocol for a (non-) DH tuple in the standard model.

Security	Unforgeability	Invisibility	Impersonation
Equivalence	CDH	$\geq$ DDH	DLOG

As a result, we successfully devise a 3-move undeniable signature scheme which is secure against active and concurrent attacks with respect to the security notions of existential unforgeability, invisibility and impersonation, with a weaker requirement than zero-knowledgeness.

Notice that schemes which adopt the non-interactive designated verifier proof technique [22] is trivially secure against impersonation attack. It is true that if a verifier has a public key, then we can make the confirmation and disavowal protocols non-interactive by using the designated verifier proof technique. However, if the verifier does not have a public key, then obviously the confirmation and disavowal protocols must remain interactive. Otherwise, non-transferability is broken.

As an aside, we remark that the idea we obtain from the observation that there exist two witnesses in a DH-tuple is of independent interest. Hopefully, it may find applications on some other interactive protocols which involve the proving of the validity of a DH-tuple in the security analyses. For example, the same idea can be readily applied to the identity-based undeniable signatures by Libert and Quisquater [23]. In [23], the scheme is proven secure using the non-interactive designated verifier proof technique only.

2 Preliminaries

Let G be an Abelian group of prime order q, and let g be a generator of G. We say that (g, g^u, g^v, g^w) is a DH-tuple if $w = uv \bmod q$.

The DDH problem is to decide if (g, g^u, g^v, g^w) is a DH-tuple. The CDH problem is to compute g^{uv} from (g, g^u, g^v) and the DLOG problem is to compute u from g^u.

2.1 The FDH Variant of Chaum's Scheme

The full domain hash (FDH) variant [14] of Chaum's undeniable signature scheme [8] is described as follows.

Let G be an Abelian group of prime order q, and let g be a generator of G.

- **Key Generation.** On input 1^k, choose $x \in Z_q$ randomly and compute $y = g^x$. Choose a cryptographic hash function $H : \{0,1\}^* \rightarrow G$. Set the public key as (g, y, H) and the secret key as x.
- **Signing.** On input the public key (g, y, H), the secret key x and a message $m \in \{0,1\}^*$, the algorithm returns the signature as $\sigma = H(m)^x$.
- **Confirmation Protocol.** Given a message-signature pair (m, σ), the signer proves that $(g, y, H(m), \sigma)$ is a DH-tuple in zero-knowledge.
- **Disavowal Protocol.** Given a pair (m, σ), the signer proves that $(g, y, H(m), \sigma)$ is not a DH-tuple in zero-knowledge.

The existing zero-knowledge confirmation protocol requires 4 moves and no zero-knowledge disavowal protocol with constant moves is known so far [8].

3 How to Formalize the Security

Traditionally, an adversary in an undeniable signature scheme intends to achieve two main adversarial goals, namely, to forge a signature and to distinguish whether a message-signature pair is valid or invalid, which correspond to the security notions of existential unforgeability and invisibility, respectively.

In this section, we introduce another new adversarial goal which is *impersonation*. We are led to this notion since our proposed confirmation/disavowal protocols are not zero-knowledge but witness indistinguishable, linking impersonation attack for identification schemes to non-transferability. Surprisingly,

this goal *impersonation* has been overlooked in the past while being different from the notions of unforgeability and invisibility.

Meanwhile, there exist three kinds of attacks, namely, passive attack, active attack and concurrent attack. We will elaborate more on this in the sequel.

3.1 Unforgeability

The first security notion is similar to the one for ordinary digital signatures, which is the notion of existential unforgeability against adaptive chosen message attack [21]. The only difference is that besides the signing oracle access, the adversary is also allowed to access to the confirmation/disavowal oracle. The confirmation/disavowal oracle is simulated based on the kind of attacks mounted, i.e. passive attack or active/concurrent attack. Generally, in a passive attack the adversary does not interact with the signer/prover. What the adversary does is eavesdropping and she is in possession of transcripts of conversations between the prover and the verifier. In an active/concurrent attack, the adversary gets to play the role of a cheating verifier, interacting with the prover several times, in an effort to extract some useful information. We will give a more formal definition shortly.

To the best of our knowledge, this is the first time passive, active and concurrent attacks are being defined explicitly and rigorously with respect to the security notions of undeniable signatures. Specifically, concurrent attack is more relevant to identification scheme [2]. However, we remark that since confirmation and disavowal protocols of an undeniable signature scheme are usually performed interactively as in an identification protocol, concurrent attack should be taken into account as well.

The difference between active and concurrent attacks is that in an active attack, the adversary interacts serially with the prover "clones"; while in a concurrent attack, the adversary is allowed to interact with many different prover "clones" concurrently. Apparently, the active/concurrent adversary has higher capability than the passive adversary.

We consider the following game.

1. Let pk be the input to a forger F.
2. The forger F is permitted to issue a series of queries:
   - Signing queries: F submits a message m and receives a signature σ on m. (We consider adaptive queries here – subsequent queries is made based on previously obtained signatures.)
   - Confirmation/disavowal queries: F submits a message-signature pair (m, σ), and the oracle responds based on whether a passive attack or an active/concurrent attack is mounted.
3. At the end of this attack game, F outputs a message-signature pair (m^*, σ^*).

- In a passive attack, the confirmation/disavowal oracle first checks the validity of (m, σ). If it is a valid pair, then the oracle returns a bit $\mu = 1$ and a transcript of confirmation protocol. Otherwise, the oracle returns a bit $\mu = 0$ and a transcript of disavowal protocol.

– In an active/concurrent attack, the confirmation/disavowal oracle first checks the validity of (m, σ). If it is a valid pair, then the oracle returns a bit $\mu = 1$ and proceeds with the execution of the confirmation protocol with the forger F (acting as a cheating verifier). Otherwise, the oracle returns a bit $\mu = 0$ and executes the disavowal protocol with F accordingly.

The forger F wins the game if F outputs a valid message-signature pair (m^*, σ^*) such that m^* has never been queried to the signing oracle, or it queries a valid (m^*, σ^*) to the confirmation/disavowal oracle such that m^* has never been queried to the signing oracle.

F's advantage in this game is defined to be $Adv(F) = \Pr[F\text{wins}]$.

Definition 1. *An undeniable signature scheme is said to be existential unforgeable under adaptive chosen message attack if no probabilistic polynomial time (PPT) forger F has a non-negligible advantage in the above game.*

3.2 Invisibility

The second security notion of undeniable signatures is invisibility, a notion due to Chaum, van Heijst and Pfitzmann [10]. This notion is essentially the inability to determine whether a given message-signature pair is valid. There are many variations in defining invisibility, for example it is defined in terms of simulatability in [10] and it is defined in terms of distinguishing whether a signature σ is corresponding to a message m_0 or m_1 in [6].

In this paper, we adopt the following definition given by Galbraith and Mao [18] as they have proven that if a scheme satisfies invisibility in the sense of Definition 2 then it also satisfies invisibility in the sense of [6].

Consider the following game.

1. Let pk be the input to a distinguisher D.
2. The distinguisher D is permitted to issue a series of queries: signing queries and confirmation/disavowal queries as in Section 3.1.
3. At some point, D outputs a message m^* which has never been queried to the signing oracle, and requests a challenge signature σ^* on m^*. The challenge signature σ^* is generated based on the outcome of a hidden coin toss b. If $b = 1$, then σ^* is generated as usual using the signing oracle, otherwise σ^* is chosen uniformly at random from the signature space S.
4. D performs some signing and confirmation/disavowal queries again with the restriction that no signing query on m^* is allowed, and no confirmation/disavowal query on the challenge message-signature pair (m^*, σ^*) is allowed.
5. At the end of this attack game, D outputs a guess b'.

The distinguisher D wins the game if $b' = b$. D's advantage in this game is defined to be $Adv(D) = |\Pr[b' = b] - \frac{1}{2}|$.

Definition 2. *An undeniable signature scheme is said to have the property of invisibility under adaptive chosen message attack if no PPT distinguisher D has a non-negligible advantage in the above game.*

The difference between the above definition and the one in [6] is such that in the latter the distinguisher D outputs two messages m_0 and m_1 and the challenge signature σ^* is generated for m_b where b is the hidden bit.

3.3 Impersonation

As noted earlier, the third (and new) security notion of undeniable signatures is the security against impersonation attack. We consider the following game.

1. Let pk be the input to an impersonator I.
2. The impersonator I enters the learning phase where it performs a series of queries: signing queries and confirmation/disavowal queries as in Section 3.1 and Section 3.2. At the end of this phase, I outputs a tuple (m^*, σ^*, μ) which consists of a message-signature pair and a bit μ (where $\mu = 1$ indicates valid and $\mu = 0$ indicates invalid).
3. In the impersonation phase, if $\mu = 1$, then the impersonator I executes the confirmation protocol with a verifier on input (m^*, σ^*). If $\mu = 0$, I executes the disavowal protocol with a verifier on input (m^*, σ^*).

The impersonator I wins the game if it can convince the verifier that (m^*, σ^*) is either valid or invalid (depending on the bit μ it outputs earlier). I's advantage in this game is defined to be $Adv(I) = \Pr[I\text{wins}]$.

Definition 3. *An undeniable signature scheme is said to be secure against impersonation under adaptive chosen message attack if no PPT impersonator I has a non-negligible advantage in the above game.*

4 WI Protocol on DH-Tuple

In this section, we present our main idea, that is, we give the descriptions of DH-tuple witness indistinguishable (WI) protocol and non DH-tuple WI protocol.

The concept of witness indistinguishability and witness hiding was introduced by Feige and Shamir [16]. Generally speaking, a two-party protocol between a prover and a verifier, in which the prover uses one of the several secret witnesses to an NP assertion, is *witness indistinguishable* if the verifier cannot tell which witness the prover is actually using. The protocol is *witness hiding* if at the end of the protocol the verifier cannot compute any new witness which he did not know before the protocol began. The result in [16] says that if a statement has at least two independent witnesses, then any witness indistinguishable protocol for this statement is also witness hiding. WI protocols have been used to construct identification schemes [16] and blind signature schemes [29, 1].

In our proposal, the prover demonstrates the knowledge of 1-out-of-2 witnesses corresponding to a problem instance (a DH-tuple) without revealing which is known, thus it is a witness indistinguishable and witness hiding protocol.

4.1 WI Protocol for DH-Tuple

Let (g, U, V, W) be a DH-tuple, where $U = g^u, V = g^v, W = g^{uv}$. Now we observe that there are two witnesses, u and v. Then by using the technique of [13], we can construct a 3-move witness indistinguishable protocol such that the prover knows u or v of a DH-tuple.

We start from a 3-move honest verifier zero-knowledge proof system (HVZK) such that the prover knows u of a DH-tuple [12]. It is depicted in Fig. 1-(a). We can obtain a similar HVZK protocol such that the prover knows v. It is symmetry to Fig. 1-(a) and thus we omit the details.

	Prover		Verifier
	$r \stackrel{R}{\leftarrow} Z_q$		
	$z_1 = g^r$		
1	$z_2 = V^r$	$\stackrel{z_1, z_2}{\longrightarrow}$	
2		$\stackrel{c}{\leftarrow}$	$c \stackrel{R}{\leftarrow} Z_q$
3	$d = r + cu \bmod q$	$\stackrel{d}{\longrightarrow}$	
			$g^d \stackrel{?}{=} z_1 U^c$
			$V^d \stackrel{?}{=} z_2 W^c$

(a) Prover knows u of a DH-tuple

	Prover		Verifier
	$r \stackrel{R}{\leftarrow} Z_q$		
	$A = (V^u / W)^r$		
	$\alpha, \beta \stackrel{R}{\leftarrow} Z_q$		
	$z_1 = V^\alpha / W^\beta$		
1	$z_2 = g^\alpha / U^\beta$	$\stackrel{A, z_1, z_2}{\longrightarrow}$	$A \stackrel{?}{\neq} 1$
2		$\stackrel{c}{\leftarrow}$	$c \stackrel{R}{\leftarrow} Z_q$
	$d_1 = \alpha + c(ur) \bmod q$		
3	$d_2 = \beta + cr \bmod q$	$\stackrel{d_1, d_2}{\longrightarrow}$	
			$V^{d_1} / W^{d_2} \stackrel{?}{=} z_1 A^c$
			$g^{d_1} / U^{d_2} \stackrel{?}{=} z_2$

(b) Prover knows u of a non DH-tuple

Fig. 1. 3-move protocols

We finally present a 3-move WI protocol such that the prover knows u or v of a DH-tuple. For this protocol, we assume that the prover knows u (but not v).

1. The prover chooses $c_2, d_2 \in Z_q$ randomly. He computes $z_1' = g^{d_2} / V^{c_2}$ and $z_2' = U^{d_2} / W^{c_2}$.
   He also chooses $r \in Z_q$ randomly and computes $z_1 = g^r$ and $z_2 = V^r$.
   Next, he sends (z_1, z_2, z_1', z_2') to the verifier.
2. The verifier chooses $c \in Z_q$ randomly and sends c to the prover.
3. The prover computes $c_1 = c - c_2 \bmod q$ and $d_1 = r + c_1 u \bmod q$. He sends (c_1, c_2, d_1, d_2) to the verifier.
4. The verifier checks if $c = c_1 + c_2 \bmod q$ and
$$g^{d_1} = z_1 U^{c_1},\ V^{d_1} = z_2 W^{c_1};$$
$$g^{d_2} = z_1' V^{c_2},\ U^{d_2} = z_2' W^{c_2}.$$

4.2 WI Protocol for Non DH-Tuple

Suppose that (g, U, V, W) is not a DH-tuple, where $U = g^u, V = g^v, W = g^w$ and $w \neq uv \bmod q$. Then similarly to Section 4.1, we can construct a 3-move WI protocol such that the prover knows u or v of a non DH-tuple.

We start from a 3-move HVZK protocol such that the prover knows u of a non DH-tuple, as proposed in [7]. The protocols is as illustrated in Fig. 1-(b). Similarly, we can obtain a 3-move HVZK protocol such that the prover knows v. It is the symmetric counterpart of Fig. 1-(b).

We finally present a 3-move WI protocol such that the prover knows u or v of a non DH-tuple. For this protocol, we assume that the prover knows u (but not v).

1. The prover chooses $c_2, d_1', d_2' \in Z_q$ randomly and $A' \in G$ such that $A' \neq 1$ randomly. He computes $z_1' = U^{d_1'}/(W^{d_2'}A'^{c_2})$ and $z_2' = g^{d_1'}/V^{d_2'}$.
   He also chooses $r \in Z_q$ randomly and computes $A = (V^u/W)^r$. Next, he chooses $\alpha, \beta \in Z_q$ randomly and computes $z_1 = V^\alpha/W^\beta$ and $z_2 = g^\alpha/U^\beta$. Finally, he sends $(A, A', z_1, z_2, z_1', z_2')$ to the verifier.
2. The verifier first checks if $A \neq 1$ and $A' \neq 1$. Next, he chooses $c \in Z_q$ randomly and sends c to the prover.
3. The prover computes $c_1 = c - c_2 \bmod q$, and $d_1 = \alpha + c_1(ur) \bmod q$ and $d_2 = \beta + c_1 r \bmod q$. He sends $(c_1, c_2, d_1, d_2, d_1', d_2')$ to the verifier.
4. The verifier checks if $c = c_1 + c_2 \bmod q$ and

$$V^{d_1}/W^{d_2} = z_1 A^{c_1},\ g^{d_1}/U^{d_2} = z_2;$$
$$U^{d_1'}/W^{d_2'} = z_1' A'^{c_2},\ g^{d_1'}/V^{d_2'} = z_2'.$$

5 Proposed 3-Move Undeniable Signature Scheme

In this section, we show a *3-move* undeniable signature scheme which is secure against active and concurrent attacks. Our scheme builds on the FDH variant of Chaum's scheme which is described earlier, by incorporating the idea from the previous section.

Since the core of this paper is to propose a 3-move undeniable signature scheme which is secure against active and concurrent attacks, we consider only security against these two kinds of attacks. Nevertheless, a scheme which is secure against active/concurrent attack will definitely secure against passive attack too.

5.1 Scheme

The key generation algorithm and the signing algorithm are the same as those of the FDH variant of Chaum's undeniable signature scheme.

(Confirmation protocol) By using the 3-move WI protocol of Section 4.1, the signer proves that $(g, y, H(m), \sigma)$ is a DH-tuple, where (m, σ) is a valid message-signature pair.

(Disavowal protocol) By using the 3-move WI protocol of Section 4.2, the signer proves that $(g, y, H(m), \sigma)$ is not a DH-tuple, where (m, σ) is not a valid message-signature pair.

5.2 Security

We show that the existential unforgeability of our proposed scheme against active and concurrent attacks is equivalent to the CDH problem in the random oracle model. Similarly, we prove that our scheme is invisible under the DDH assumption and the impersonation is equivalent to the DLOG problem.

Theorem 1. *The existential unforgeability of the above 3-move undeniable signature scheme against active and concurrent attacks is equivalent to the CDH problem in the random oracle model.*

Proof. Please refer to Appendix A. □

Theorem 2. *The above 3-move undeniable signature scheme is invisible against active and concurrent attacks under the DDH assumption in the random oracle model.*

Proof. Please refer to Appendix B. □

Theorem 3. *The security against impersonation under active and concurrent attacks of the above 3-move undeniable signature scheme is equivalent to the DLOG problem in the random oracle model.*

Proof. Please refer to Appendix C. □

6 Discussion

A naive approach for witness indistinguishability would be to use two public keys, where the two corresponding secret keys are two witnesses, as suggested in [16]. However, this approach does not work as shown below.

In this approach, the signer has two public keys, $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$. The undeniable signature on a message m is $\sigma = (\sigma_1, \sigma_2)$, where $\sigma_1 = H(m)^{x_1}$ and $\sigma_2 = H(m)^{x_2}$. The secret key of the signer is x_1 or x_2. In the confirmation protocol, the signer proves that σ_1 is valid OR σ_2 is valid.

However, in the disavowal protocol, the signer has to prove that "σ_1 is invalid AND σ_2 is invalid" because De Morgan's law claims that $\overline{X \vee Y} = \overline{X} \wedge \overline{Y}$. Therefore, the disavowal protocol cannot be witness indistinguishable. In general, from De Morgan's law, both the confirmation protocol and the disavowal protocol cannot be witness indistinguishable simultaneously. On the other hand, we manage to circumvent this problem by our new approach.

We further remark that our proposed scheme is almost as efficient as the FDH variant of Chaum's scheme, except that the computation and communication complexity in the confirmation and disavowal protocols are almost twice the original scheme. However, we stress that this slight efficiency loss is worthwhile in achieving the security against active and concurrent attacks with only *3-move* confirmation and disavowal protocols. This is indeed a significant contribution to the literature of undeniable signatures.

7 Conclusion

We proposed the first 3-move undeniable signature scheme which is provably secure against active and concurrent attacks, by exploiting the fact that DH-tuples possess two witnesses, and also that non DH-tuples possess two witnesses. Thus, this allows us to use the concept of witness indistinguishability and witness hiding in the confirmation and disavowal protocols of the FDH variant of Chaum's scheme. The existential unforgeability of our proposed scheme against adaptive chosen message attack is equivalent to the CDH problem. The scheme satisfies the property of invisibility assuming the intractability of the DDH problem. Moreover, we also introduced another security notion which is impersonation attack. We proved that the security against impersonation of our proposed scheme is equivalent to the DLOG problem.

References

1. M. Abe and T. Okamoto. Provably secure partially blind signatures. *Advances in Cryptology — CRYPTO '00*, LNCS 1880, pp. 271–286, Springer-Verlag, 2000.
2. M. Bellare and A. Palacio. GQ and Schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks. *Advances in Cryptology — CRYPTO '02*, LNCS 2442, pp. 162–177, Springer-Verlag, 2002.
3. J. Boyar, D. Chaum, I. Damgård and T. Pedersen. Convertible undeniable signatures. *Advances in Cryptology — CRYPTO '90*, LNCS 537, pp. 189–208, Springer-Verlag, 1990.
4. C. Boyd and E. Foo. Off-line fair payment protocols using convertible signatures. *Advances in Cryptology — ASIACRYPT '98*, LNCS 1514, pp. 271–285, Springer-Verlag, 1998.
5. I. Biehl, S. Paulus and T. Takagi. Efficient undeniable signature schemes based on ideal arithmetic in quadratic orders. *Designs, Codes and Cryptography*, Vol. 31, Issue 2, pp. 99–123, 2004
6. J. Camenisch and M. Michels. Confirmer signature schemes secure against adaptive adversaries. *Advances in Cryptology — EUROCRYPT '00*, LNCS 1870, pp. 243–258, Springer-Verlag, 2000.
7. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. *Advances in Cryptology — CRYPTO '03*, LNCS 2729, pp. 126–144, Springer-Verlag, 2003.
8. D. Chaum. Zero-knowledge undeniable signatures. *Advances in Cryptology — EUROCRYPT '90*, LNCS 473, pp. 458–464, Springer-Verlag, 1990.
9. D. Chaum. Designated confirmer signatures. *Advances in Cryptology — EUROCRYPT '94*, LNCS 950, pp. 86–91, Springer-Verlag, 1995.
10. D. Chaum, E. van Heijst and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. *Advances in Cryptology — CRYPTO '91*, LNCS 576, pp. 470–484, Springer-Verlag, 1991.
11. D. Chaum and H. van Antwerpen. Undeniable signatures. *Advances in Cryptology — CRYPTO '89*, LNCS 435, pp. 212–216, Springer-Verlag, 1989.
12. T. Chaum and T. P. Pedersen. Wallet databases with observers. *Advances in Cryptology — CRYPTO '92*, LNCS 740, pp. 89–105, Springer-Verlag, 1993.

13. R. Cramer, I. Damgård and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *Advances in Cryptology — CRYPTO '94*, LNCS 839, pp. 174–187, Springer-Verlag, 1994.
14. J. Coron. On the exact security of full domain hash. *Advances in Cryptology — CRYPTO '00*, LNCS 1880, pp. 229–235, Springer-Verlag, 2000.
15. I. Damgård and T. Pedersen. New convertible undeniable signature schemes. *Advances in Cryptology — EUROCRYPT '96*, LNCS 1070, pp. 372–386, Springer-Verlag, 1996.
16. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. *ACM Symposium on Theory of Computing — STOC '90*, pp. 416–426, 1990.
17. L. Fortnow. The complexity of perfect zero-knowledge (extended abstract). *ACM Symposium on Theory of Computing — STOC '87*, pp. 204-209, 1987.
18. S. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. *Topics in Cryptology — CT-RSA '03*, LNCS 2612, pp. 80–97, Springer Verlag, 2003.
19. S. Galbraith, W. Mao and K. G. Paterson. RSA-based undeniable signatures for general moduli. *Topics in Cryptology — CT-RSA '02*, LNCS 2271, pp. 200–217, Springer Verlag, 2002.
20. R. Gennaro, H. Krawczyk and T. Rabin. RSA-based undeniable signatures. *Advances in Cryptology — CRYPTO '97*, LNCS 1294, pp. 132–149, Springer-Verlag, 1997.
21. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptative chosen-message attacks. *SIAM Journal of Computing*, vol. 17, no. 2, pp. 281–308, 1988.
22. M. Jakobsson, K. Sako and R. Impagliazzo. Designated verifier proofs and their applications. *Advances in Cryptology — EUROCRYPT '96*, LNCS 1070, pp. 143–154, Springer-Verlag, 1996.
23. B. Libert and J.-J Quisquater. Identity based undeniable signatures. *Topics in Cryptology — CT-RSA '04*, LNCS 2964, pp. 112–125, Springer-Verlag, 2004.
24. M. Michels and M. Stadler. Efficient convertible undeniable signature schemes. *Selected Areas in Cryptography — SAC '97*, pp. 231–244, Springer-Verlag, 1997.
25. J. Monnerat and S. Vaudenay. Undeniable signatures based on characters: how to sign with one bit. *Public Key Cryptography — PKC'04*, LNCS 2947, pp. 361–396, Springer-Verlag, 2004.
26. J. Monnerat and S. Vaudenay. Generic homomorphic undeniable signatures. *Advances in Cryptology — Asiacrypt '04*, LNCS 3329, pp. 354–371, Springer-Verlag, 2004.
27. T. Okamoto. Designated confirmer signatures and public key encryption are equivalent. *Advances in Cryptology — CRYPTO '94*, LNCS 839, pp. 61–74, Springer-Verlag, 1994.
28. D. Pointcheval. Self-scrambling anonymizers. *Financial Cryptography — FC '00*, LNCS 1962, pp. 259–275, Springer-Verlag, 2000.
29. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, Springer-Verlag, 2000.

A Proof of Theorem 1

Firstly, we show that if there exists an algorithm M that solves the CDH problem with advantage ϵ_M, then one can construct a forger F that can forge in the

universal way with advantage ϵ_F, by running M as a subroutine. The forger F is given the public key (g, y, H) where $y = g^x$. For any message m, F computes $h = H(m)$ and gives the triple (g, y, h) as input to M. When M outputs h^x, F simply outputs the forgery as $(m, \sigma = h^x)$. It is clear that $\epsilon_F = \epsilon_M$. This completes the first half of our proof.

Secondly, we show that if there exists an existential forger F with advantage ϵ_F, then one can construct an algorithm M that can solve the CDH problem with advantage ϵ_M, by running F as a subroutine. Suppose the input to M is (g, g^x, g^z). M then starts running F by feeding F with the public key $(g, y = g^x, H)$ where H is a random oracle that will be simulated by M. M also simulates the signing oracle and the confirmation/disavowal oracle itself. Let q_S and q_H be the number of signing queries and H queries that F issues respectively. We assume that when F requests a signature on a message m_i, it has already made the corresponding H query on m_i.

When F makes a H query for a message m_i, M responds with $h_i = H(m_i) = g^{v_i}$ with probability δ and $h_i = H(m_i) = (g^z)^{v_i}$ with probability $1 - \delta$, where v_i is chosen randomly from Z_q and δ is a fixed probability which will be determined later. Suppose that F makes a signing query for a message m_i. If M has responded with $h_i = g^{v_i}$ to the H query for a message m_i, then M returns $\sigma_i = y^{v_i}$ as the valid signature (since $y^{v_i} = (g^x)^{v_i} = h_i^x = H(m_i)^x$). Otherwise, M aborts and it fails to solve the CDH problem.

Next, we consider the case when F makes a confirmation/disavowal query. Let q_v be the number of queries that F issues to the confirmation/disavowal oracle. For convenience, we consider that the final output of F is the $(q_v + 1)$th query. We say that (m_i, σ_i') is special if it is a valid message-signature pair queried by F to the confirmation/disavowal oracle such that m_i has never been queried to the signing oracle. M guesses the first special query. More precisely, M guesses the first i such that the ith query (m_i, σ_i') is special. So, at the beginning, M chooses $Guess \in \{1, 2, \cdots, q_v + 1\}$ randomly. There are two cases to be considered here, namely, $i < Guess$ and $i = Guess$. First suppose that $i < Guess$.

- If F has never made a signing query for m_i, then M returns $\mu = 0$ and runs the disavowal protocol with F.
- Otherwise, F has already made a signing query for m_i, and M answered with a valid signature σ_i with probability δ (with probability $(1 - \delta)$ M aborts). If $\sigma_i = \sigma_i'$ then M returns $\mu = 1$ and runs the confirmation protocol with F. Otherwise, M returns $\mu = 0$ and runs the disavowal protocol with F.

Notice that since M knows v_i, it can simulate the confirmation/disavowal oracle perfectly. (Recall that the execution of the confirmation/disavowal oracle is to prove whether $(g, g^x, H(m_i) = g^{v_i}, H(m_i)^x = g^{v_i x})$ is a DH-tuple or not. Since M knows one of the witnesses which is v_i, it can simulate the interactive proof perfectly.)

Now suppose that $i = Guess$. Let (m^*, σ^*) be the ith query. If F has queried m^* to the signing oracle, then M aborts. Otherwise, we assume that F has queried the H-oracle on m^* and so $m^* = m_j$ for some j. If $h_j = (g^z)^{v_j}$, then

we have $\sigma^* = h_j^x = (g^{zv_j})^x$. Consequently, M outputs $g^{xz} = (\sigma^*)^{1/v_j}$ and thus it solves the CDH problem. Otherwise, M aborts and it fails to solve the CDH problem.

To complete the proof, it remains to calculate the probability that M does not abort. M guesses the first special query with probability $1/(q_v + 1)$. The probability that M answers to all the signing queries is δ^{q_S} and M outputs g^{zr} with probability $1 - \delta$. Therefore, the probability that M does not abort during the simulation is $\delta^{q_S}(1-\delta)/(q_v+1)$. This value is maximized at $\delta_{opt} = 1-1/(q_S+1)$. This shows that M's advantage ϵ_M is at least $(1/e(1+q_S))\epsilon_F/(q_v+1)$, where e is the base of the natural logarithm. This is because the value $(1-1/(q_S+1))^{q_S}$ approaches $1/e$ for large q_S. This completes our proof.

B Proof of Theorem 2

We show that if there exists an invisibility distinguisher D with advantage ϵ_D, then one can construct an DDH distinguisher D' with advantage $\epsilon_{D'}$, by running D as a subroutine. Suppose the input to D' is (g, g^x, g^z, g^t). D' then starts running D by feeding D with the public key $(g, y = g^x, H)$ where H is a random oracle that will be simulated by D'. D' also simulates the signing oracle and the confirmation/disavowal oracle itself. Let q_S and q_H be the number of signing queries and H queries that D issues respectively. We assume that when D requests a signature on a message m_i, it has already made the corresponding H query on m_i.

When D makes a H query for a message m_i, D' responds with $h_i = H(m_i) = g^{v_i}$ with probability δ and $h_i = H(m_i) = (g^z)^{v_i}$ with probability $1 - \delta$, where v_i is chosen randomly from Z_q and δ is a fixed probability which will be determined later. Suppose that D makes a signing query for a message m_i. If D' has responded with $h_i = g^{v_i}$ to the H query for a message m_i, then D' returns $\sigma_i = y^{v_i}$ as the valid signature (since $y^{v_i} = (g^x)^{v_i} = h_i^x = H(m_i)^x$). Otherwise, D' aborts and it fails to solve the DDH problem.

Eventually, D outputs a message m^*. We assume that D has queried the H-oracle on m^* and so $m^* = m_i$ for some i. If $h_i = (g^z)^{v_i}$, then D' returns the challenge signature $\sigma = (g^t)^{v_i}$. Otherwise, D' aborts and it fails to solve the DDH problem.

Next, D performs some H queries, signing queries and confirmation/disavowal queries again with the restriction that no signing queries on m^* is allowed, and no confirmation/disavowal query on the challenge message-signature pair (m^*, σ^*) is allowed.

Finally, D outputs a bit b' which it thinks is equal to the hidden bit b. More precisely, D outputs $b' = 1$ if it finds that (m^*, σ^*) is a valid message-signature pair and it outputs $b' = 0$ if it finds that σ^* is chosen uniformly at random from the signature space S.

Subsequently, D' provides the same output as D which is b'. Note that if (m^*, σ^*) is a valid message-signature pair, then (g, g^x, g^z, g^t) is a DH-tuple. This is indeed the case since $\sigma^* = h_i^x$ implies that $t = xz \bmod q$, where $\sigma^* = (g^t)^{v_i}$

and $h_i = (g^z)^{v_i}$. Otherwise (g, g^x, g^z, g^t) is not a DH-tuple. This is indeed the case since $\sigma^* \neq h_i^x$ implies that $t \neq xz \bmod q$. Therefore, if D is an invisibility distinguisher then D' is a DDH distinguisher.

Now, we show how to simulate the confirmation/disavowal oracle. If CDH problem is easy, then DDH problem is easy. Hence D' can solve the DDH problem (without using D) in this case.

Suppose that CDH problem is hard. Then D cannot forge (m_i, σ_i) with non-negligible probability because forgery is equivalent to CDH problem from Theorem 1. Now assume that D queries (m_i, σ_i') to the confirmation/disavowal oracle.

- If D has never made a signing query for m_i, then D' returns $\mu = 0$ and runs the disavowal protocol with D. This is justified because D cannot forge as mentioned above.
- Otherwise, D has already made a signing query for m_i, and D' has answered with a valid signature σ_i. If $\sigma_i = \sigma_i'$ then D' returns $\mu = 1$ and runs the confirmation protocol with D. Otherwise, D' returns $\mu = 0$ and runs the disavowal protocol with D.

(M can run the confirmation/disavowal protocol as in the proof of Theorem 1.)

To complete the proof, it remains to calculate the probability that D' does not abort. The probability that D' answers to all the signing queries is δ^{q_S} and D' succeeds in distinguishing the DDH problem with probability $1-\delta$. Therefore, the probability that D' does not abort during the simulation is $\delta^{q_S}(1-\delta)$. This value is maximized at $\delta_{opt} = 1 - 1/(q_S + 1)$. This shows that D''s advantage $\epsilon_{D'}$ is at least $(1/e(1+q_S))\epsilon_D$, where e is the base of the natural logarithm. This is because the value $(1-1/(q_S+1))^{q_S}$ approaches $1/e$ for large q_S. This completes our proof.

C Proof of Theorem 3

Firstly, we show that if there exists an algorithm M that solves the DLOG problem with advantage ϵ_M, then one can construct an impersonator I that can succeed in an impersonation by running M as a subroutine, with advantage ϵ_I. At first, the impersonator I is given the public key (g, y, H) where $y = g^x$. Since I can obtain the secret key x by feeding y to the algorithm M, it can impersonate the signer with the knowledge of x. It is clear that $\epsilon_I = \epsilon_M$. This completes the first half of our proof.

Secondly, we show that if there exists an impersonator I with advantage ϵ_I, then one can construct an algorithm M that can solve the DLOG problem with advantage ϵ_M, by running I as a subroutine. Suppose the input to M is (g, g^x). M first chooses a bit coin.

Suppose that $\mathsf{coin} = 0$. M then starts running I by feeding I with the public key $(g, y = g^x, H)$ where H is a random oracle that will be simulated by M. M also simulates the signing oracle and the confirmation/disavowal oracle itself. We assume that when I requests a signature on a message m_i, it has already made the corresponding H query on m_i.

In the learning phase, I starts a series of queries. When I makes a H query for a message m_i, M responds with $h_i = H(m_i) = g^{v_i}$, where v_i is chosen randomly from Z_q. When I makes a signing query for a message m_i, M returns $\sigma_i = y^{v_i}$ as the valid signature (since $y^{v_i} = (g^x)^{v_i} = h_i^x = H(m_i)^x$).

Suppose that I makes a confirmation/disavowal query for a message-signature pair (m_i, σ_i'). If m_i has never been queried to the signing oracle by I, then M simulates the signing oracle as above by itself. Hence M knows a valid signature σ_i anyway. Then M returns $\mu = 1$ if $\sigma_i' = \sigma_i$ and $\mu = 0$ if $\sigma_i' \neq \sigma_i$. M also runs the confirmation or disavowal protocol accordingly, where M can run the confirmation/disavowal protocol as in the proof of Theorem 1.

At the end of this learning phase, I outputs a tuple (m^*, σ^*, μ).

Next, I enters the impersonation phase. If $\mu = 1$, then I executes the confirmation protocol with M (acting as a verifier) on input (m^*, σ^*). M runs I to obtain its commitment (z_1, z_2, z_1', z_2'), randomly selects a challenge $c \in Z_q$, and runs I to obtain its response (c_1, c_2, d_1, d_2). M next resets I to the step whereby I has sent (z_1, z_2, z_1', z_2'). M then randomly selects a fresh challenge $c' \in Z_q$, and re-runs I to obtain its response (c_1', c_2', d_1', d_2').

If both conversations are accepted and $c \neq c'$, then M can extract the DLOG of y (which is x) or the DLOG of $H(m^*)$ (which is $v = v_i$ for some v_i) as follows. Before this, remember that $c = c_1 + c_2 \bmod q$ and $c' = c_1' + c_2' \bmod q$. This implies that $c_1 \neq c_1'$ or $c_2 \neq c_2'$, otherwise $c_1 = c_2$ which contradicts the above assumption.

From the first conversation, we obtain

$$g^{d_1} = z_1 y^{c_1}, \quad H(m^*)^{d_1} = z_2 (\sigma^*)^{c_1};$$
$$g^{d_2} = z_1' H(m^*)^{c_2}, \quad y^{d_2} = z_2' (\sigma^*)^{c_2}.$$

From the second conversation, we obtain

$$g^{d_1'} = z_1 y^{c_1'}, \quad H(m^*)^{d_1'} = z_2 (\sigma^*)^{c_1'};$$
$$g^{d_2'} = z_1' H(m^*)^{c_2'}, \quad y^{d_2'} = z_2' (\sigma^*)^{c_2'}.$$

Then it is not difficult to see that

$$g^{d_1 - d_1'} = y^{c_1 - c_1'}, \quad H(m^*)^{d_1 - d_1'} = (\sigma^*)^{c_1 - c_1'}; \tag{1}$$
$$g^{d_2 - d_2'} = H(m^*)^{c_2 - c_2'}, \quad y^{d_2 - d_2'} = (\sigma^*)^{c_2 - c_2'}. \tag{2}$$

When $c_1 \neq c_1'$, since $y = g^x$ and $\sigma^* = H(m^*)^x$, M can extract $x = \frac{d_1 - d_1'}{c_1 - c_1'} \bmod q$ from (1). When $c_2 \neq c_2'$, M can extract $v = \frac{d_2 - d_2'}{c_2 - c_2'} \bmod q$ from (2).

On the other hand, if $\mu = 0$, then the impersonator I executes the disavowal protocol with M (acting as a verifier) on input (m^*, σ^*). M runs I to obtain its commitment $(A, A', z_1, z_2, z_1', z_2')$, randomly selects a challenge $c \in Z_q$, and runs I to obtain its response $(c_1, c_2, d_{11}, d_{12}, d_{11}', d_{12}')$. M next resets I to the step whereby I has sent $(A, A', z_1, z_2, z_1', z_2')$. M then randomly selects a fresh challenge $c' \in Z_q$, and re-runs I to obtain its response $(c_1', c_2', d_{21}, d_{22}, d_{21}', d_{22}')$.

Again, if both conversations are accepted and $c \neq c'$, then M can extract the DLOG of y (which is x) or the DLOG of $H(m^*)$ (which is v) as follows. With the same argument as above, since $c = c_1 + c_2 \bmod q$ and $c' = c'_1 + c'_2 \bmod q$, this implies that $c_1 \neq c'_1$ or $c_2 \neq c'_2$.

From the first conversation, we obtain

$$H(m^*)^{d_{11}}(\sigma^*)^{-d_{12}} = z_1 A^{c_1}, \quad g^{d_{11}} y^{-d_{12}} = z_2;$$
$$y^{d'_{11}}(\sigma^*)^{-d'_{12}} = z'_1 A'^{c_2}, \quad g^{d'_{11}} H(m^*)^{-d'_{12}} = z'_2.$$

From the second conversation, we obtain

$$H(m^*)^{d_{21}}(\sigma^*)^{-d_{22}} = z_1 A^{c'_1}, \quad g^{d_{21}} y^{-d_{22}} = z_2;$$
$$y^{d'_{21}}(\sigma^*)^{-d'_{22}} = z'_1 A'^{c'_2}, \quad g^{d'_{21}} H(m^*)^{-d'_{22}} = z'_2.$$

From the above equations, we would obtain

$$H(m^*)^{d_{11}-d_{21}}(\sigma^*)^{-(d_{12}-d_{22})} = A^{c_1-c'_1}, \quad g^{d_{11}-d_{21}} y^{-(d_{12}-d_{22})} = 1; \tag{3}$$
$$y^{d'_{11}-d'_{21}}(\sigma^*)^{-(d'_{12}-d'_{22})} = A'^{c_2-c'_2}, \quad g^{d'_{11}-d'_{21}} H(m^*)^{-(d'_{12}-d'_{22})} = 1. \tag{4}$$

When $c_1 \neq c'_1$, since $y = g^x$ and $A = (H(m^*)^x/(\sigma^*))^r$, M can extract $x = \frac{d_{11}-d_{21}}{d_{12}-d_{22}} \bmod q$ from (3). When $c_2 \neq c'_2$, since $y = g^x$ and $A' = ((g^x)^v/(\sigma^*))^r$, M can extract $v = \frac{d'_{11}-d'_{21}}{d'_{12}-d'_{22}} \bmod q$ from (4).

Finally, for both confirmation and disavowal protocols, by Reset Lemma [2], the probability that algorithm M accepts both conversations and that $c \neq c'$ is at least $(\epsilon_I - \frac{1}{q})^2$. This shows that M can extract the DLOG of y (which is x) or the DLOG of $H(m^*)$ (which is $v = v_i$ for some v_i) with probability at least $(\epsilon_I - \frac{1}{q})^2$.

Suppose that $\mathsf{coin} = 1$. In this case, M behaves as above with the modifications as follows: M chooses $\alpha \in Z_q$ randomly, and let $y = g^\alpha$, $H(m_i) = (g^x)^{v_i}$ and $\sigma_i = (g^x)^{\alpha v_i}$, where v_i is chosen randomly from Z_q. Finally, M can extract the DLOG of y (which is α) or the DLOG of $H(m^*)$ (which is xv_i for some v_i) with probability at least $(\epsilon_I - \frac{1}{q})^2$ as in the case of $\mathsf{coin} = 0$.

This means that M's advantage in extracting x is at least $\frac{1}{2}(\epsilon_I - \frac{1}{q})^2$ because I has no information on coin. This completes our proof.

Group Signatures with Efficient Concurrent Join

Aggelos Kiayias[1,**] and Moti Yung[2]

[1] Computer Science and Engineering, University of Connecticut
Storrs, CT, USA
aggelos@cse.uconn.edu
[2] RSA Laboratories, Bedford, MA, and
Computer Science, Columbia University
New York, NY, USA
moti@cs.columbia.edu

Abstract. A group signature is a basic privacy mechanism. The group joining operation is a critical component of such a scheme. To date all secure group signature schemes either employed a trusted-party aided join operation or a complex joining protocol requiring many interactions between the prospective user and the Group Manager (GM). In addition no efficient scheme employed a join protocol proven secure against adversaries that have the capability to dynamically initiate multiple concurrent join sessions during an attack.

This work presents the first efficient group signature scheme with a simple Joining protocol that is based on a "single message and signature response" interaction between the prospective user and the GM. This single-message and signature-response registration paradigm where no other actions are taken, is the most efficient possible join interaction and was originally alluded to in 1997 by Camenisch and Stadler, but its efficient instantiation remained open till now.

The fact that joining has two short communication flows and does not require secure channels is highly advantageous: for example, it allows users to easily join by a proxy (i.e., a security officer of a company can send a file with all registration requests in his company and get back their certificates for distribution back to members of the company). It further allows an easy and non-interactive global system re-keying operation as well as straightforward treatment of multi-group signatures. We present a strong security model for group signatures (the first explicitly taking into account concurrent join attacks) and an efficient scheme with a single-message and signature-response join protocol.

1 Introduction

Group signatures is a useful anonymous non-repudiable multi-use credential primitive that was introduced by Chaum and Van Heyst [17]. It involves a group

** Research partly supported by NSF CAREER Award CNS-0447808.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 198–214, 2005.

of users, each holding a membership certificate that allows a user to issue a publicly verifiable signature while hiding the identity of the actual signer within the group. The public-verification procedure employs only the public-key of the group. Furthermore, in the case of any dispute or abuse, it is possible for the group manager (GM) to "open" an individual signature and reveal the identity of its originator.

Constructing an efficient group signature has been a research target for many years, see e.g., [18, 16, 13, 14, 9, 26, 3, 1, 11, 24, 8, 2, 10, 12]. A scalable scheme that provides constant signature size and has resistance to attacks by coalitions of users was given in [1]. Earlier constructions were designed without a formal model and definition of security of such schemes, and thus with partial security proofs at the best case (while many were actually broken).

A central issue in group signatures has been the way by which users join the group. Recently, [5] gave the first formal model of a somewhat "relaxed" group signature primitive where a trusted party generates and hands out all users' keys. They also produced a generic solution thus demonstrating the polynomial-time plausibility of their notion of trusted-party aided join group signatures. This is in contrast with users who dynamically join the system and get their individual keys by interacting with the group manager (as in the protocol of [1]). Dynamic joins that allow users to register sequentially were studied formally in [23, 25] where efficient constructions were given and in [5, 6] where a generic plausibility proof was provided.

The most efficient and conceptually simple joining procedure for a group signature scheme (what we will call the "single-message and signature-response paradigm") was illustrated by Camenisch and Stadler [16] who sketched a generic solution (which was followed in careful details in [5, 6]). In this type of joining protocol, the prospective user has an appropriately distributed secret x' and it computes a one way function f on it to obtain $x = f(x')$. The user sends x to the GM who, in turn, signs x and returns the signature v to the user using an appropriate signing algorithm. This completes the interaction of the join protocol. The possession ofthe signature v on $x = f(x')$ enables a user to sign anonymously a message m by simply encrypting x probabilistically into ψ (under the GM's public key or whatever entity is supposed to execute the opening algorithm) and by providing a zero-knowledge proof of (i) the fact that the ψ is an encryption of some x known to the prover, (ii) the fact that the prover knows x' a preimage of that x under f, (iii) the fact that the prover knows a signature issued by the GM on that x.

While the Camenisch-Stadler approach is elegant and advantageous (as we argue below), its instantiation by an efficient scheme turned out to be elusive, since the many schemes that have been suggested in the last eight years approximated it but none really employed it. In fact, all the efficient schemes in the non-trusted-party-aided joining setting that were not broken used additional communications during the join protocol usually to assure that certain constraints and certain knowledge of the joining user is present, i.e., the prospective user had to engage in an interactive zero-knowledge proof with the GM. It was not at all appar-

ent whether the single-message and signature-response join would actually be instantiable in an *efficient* manner in a *provably secure scheme*. Moreover the employment of such proofs of knowledge has the usual shortcomings with respect to adversaries operating in the concurrent setting (namely, rewinding cannot be employed and a "straight-line" approach needs to be followed that makes the joining protocols even more involved).

To conclude the motivation for our result, we summarize the advantages of a group signature employing a single-message and signature-response joining protocol:

1. *Concurrency:* Joining of users can be done concurrently where a batch of users join at the same time. This enables group managers over the Internet (where servers are multi-thread machines).
2. *Proxy Join:* Users can be joined by a proxy collecting all their requests and then collecting the responses from the group manager; this is a very effective way to enroll companies and organizations by delegating collection and distribution to security officers. It is highly effective in enrolling to an identity escrow scheme without the need for random oracle proofs.
3. Multi-Group Scenario: There may be a number of groups; since single-message and signature-response joins require essentially no interaction between the GM and the prospective user users may accumulate many GM membership signatures on the same x value non-interactively thus easily becoming members of multiple-groups.

1.1 Our Result

In this work we implement the first group signature scheme with a single-message and signature-response join protocol to be exploited for concurrent joins and other advantages as above, thus implementing efficiently the Camenisch-Stadler approach for the first time[1].

We start by presenting the first model of "group signature with concurrent joins" which builds on the recent formal models and consists of a set of attacks. We note that in a privacy primitive interacting users may be conducting simultaneous attacks against each other and these need to be captured formally. We call our attacks: misidentification attack, framing attack and anonymity attack and is an extension of our sequential-join formal model for group signatures in [25]. We then implement a scheme based on specific assumptions and prove its security. The scheme allows adversarial opening of signatures and its signature size is only about twice the size of the scheme of [1] (that did not allow for adversarial opening or concurrent join attacks).

[1] In some recent schemes of group signatures and related primitives based on dynamic accumulators [28, 19], a simple two message join was implemented; nevertheless this was to be followed by local modifications of keys of *all existing users*; we do not consider such a protocol efficient. In our solution, keys of other users are unaffected when new members are introduced to the group.

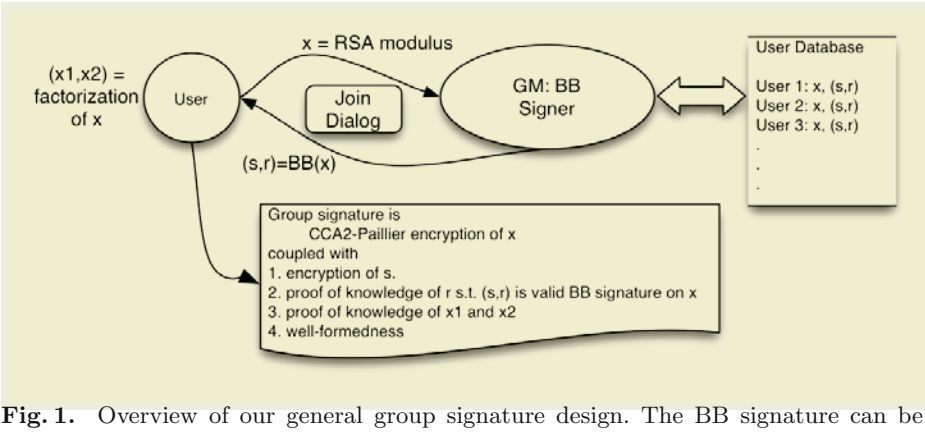

Fig. 1. Overview of our general group signature design. The BB signature can be substituted by potentially other signatures that are suitable for algebraic encryption with efficient validity proof

From a technical viewpoint we employ a number of complex primitives including the digital signature scheme of Boneh and Boyen [7] (hence referred to as the BB signature) as well as verifiable encryption for discrete-logarithms that are based on the Paillier encryption function [27, 20, 15, 22].

A novelty of our technical approach (and perhaps an explanation why we manage to achieve an efficient single-message and signature-response join) is that we deviate from most of group signature literature by instantiating the one-way function employed by the prospective user during the join with multiplication instead of exponentiation. Our general design approach is outlined in figure 1: users sample an RSA modulus and merely obtain a BB certificate on it. This modest interaction (which is simply a PKI registration in a domain employing RSA moduli with a BB signature for certification) allows users to sign as group members.

Our security proofs follow a modular approach: in a nutshell, a misidentification adversary is turned into a BB-forger, a framing adversary is turned into a factoring algorithm and an anonymity attacker is turned into a CCA2 adversary against the encryption algorithm we employ. The group signature itself is based on the Fiat-Shamir paradigm, by essentially turning an identity escrow (anonymous identification) system into a signature and employing a random oracle. We note that the interactive version of our group signature yields an identity escrow scheme in a straightforward manner that can also have concurrent group signing by employing general transformation techniques for Σ-protocols, e.g. [21].

2 Preliminaries

Interactive Turing Machines and Concurrent Executions. A two-party protocol is a pair of probabilistic polynomial-time bounded Interactive Turing machines $\langle \mathsf{A}, \mathsf{B} \rangle$. Each of A, B has a private input tape, work-tapes, a (joint)

communication tape and a private output tape. An execution of a protocol $\langle \mathsf{A}, \mathsf{B} \rangle$ on inputs x, y for the two players will be denoted by $[\mathsf{A}(x), \mathsf{B}(y)]$. For an execution of a protocol we will consider the following random variables: (i) $\mathsf{Trans}[\mathsf{A}(x), \mathsf{B}(y)]$ is the contents of the communication tape after the two parties terminate. (ii) $\mathsf{Out}_{\mathsf{A}}[\mathsf{A}(x), \mathsf{B}(y)]$ is the contents of the private output tape of player A after termination. (iii) $\mathsf{Out}_{\mathsf{B}}[\mathsf{A}(x), \mathsf{B}(y)]$ is the contents of the private output tape of player B after termination.

Now suppose that $\mathcal{P} = \langle \mathsf{A}, \mathsf{B} \rangle$ is a protocol. An "interface oracle" for concurrent simulation of player B, denoted by $\mathcal{I}[\mathcal{P}_{\leftrightarrow \mathsf{B}(y)}]$, is an oracle that accepts the following queries:

Q1. $\texttt{Start} - \texttt{Session}$: The interface oracle $\mathcal{I}[\mathcal{P}_{\leftrightarrow \mathsf{B}(y)}]$ initiates a session for the protocol $\mathcal{P}$: it selects a session identifier s and if B is the player that goes first in the protocol $\mathcal{P}$, the interface simulates the first move of B on input y; the interface returns as answer to the $\texttt{Start} - \texttt{Session}$ query the session identifier s and the output of the simulation of player B's first move (if any). The interface keeps a database with the state of player B for the session identifier s; the state includes all coin tosses of B, and the contents of all tapes including the communication tape.

Q2. $\texttt{Advance} - \texttt{Session}(s, msg)$: The interface oracle looks up the table of sessions and recovers the state of player B for the session with identifier s (if there is no such session the interface returns $\perp$ as answer to the oracle query). If session s exists the interface appends msg to the communication tape of the session and continues the simulation of player B (as if msg is the message that is written to the communication tape of player B by player A).

We will use the notation $M^{\mathcal{I}[\mathcal{P}_{\leftrightarrow \mathsf{B}(\cdot)}]}$ to denote any probabilistic Turing machine M that has access to an interface oracle as defined above. Note that the interface oracle $\mathcal{I}[\mathcal{P}_{\leftrightarrow \mathsf{A}(x)}]$ (for concurrent executions of player A in the protocol $\mathcal{P}$) can be defined in the same fashion as above. Frequently protocol executions are stateful, e.g. there is a database, or state St in general that an instantiation of the protocol $\mathcal{P}$ may consult. This state St will be maintained by the interface oracle $\mathcal{I}$. In this case we will write $\mathcal{I}_{St}[\mathcal{P}_{\leftrightarrow \mathsf{B}(\cdot)}]$. In the case that a TM M has access to a stateful interface oracle $\mathcal{I}$ we will write $M^{\mathcal{I}_{St}[\mathcal{P}_{\leftrightarrow \mathsf{B}(\cdot)}]}$. Depending on the case, $\mathcal{I}$ may modify the state St or even allow read and write access to St by M.

Bilinear Maps. Let $\mathbb{G}_1, \mathbb{G}_2$ two groups of prime order p so that (i) $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$; (ii) $\tau : \mathbb{G}_2 \to \mathbb{G}_1$ is an isomorphism with $\tau(g_2) = g_1$ and (iii) $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. We remark that in many cases it can be that $\mathbb{G}_1 = \mathbb{G}_2$ (and in this case ψ would be the identity mapping). Let $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle$ groups as above with $|\mathbb{G}_1| = |\mathbb{G}_2| = p$; a bilinear map is a map e s.t. for all $(u, v) \in \mathbb{G}_1 \times \mathbb{G}_2$ it holds that $e(u^x, v^y) = e(u, v)^{xy}$ and $e(g_1, g_2) \neq 1$.

Intractability Assumptions. We will employ the following four intractability assumptions:

The *Strong Diffie Hellman Assumption* (SDH) was put forth by Boneh and Boyen [7]. The q-SDH problem over two groups $\mathbb{G}_1, \mathbb{G}_2$ is defined as follows: given a $(q+2)$-tuple $\langle g_1, g_2, g_2^{\gamma}, \ldots, g_2^{(\gamma)^q} \rangle$ as input, output a pair $(g_1^{\frac{1}{\gamma+x}}, x)$ where $x \in \mathbb{Z}_p^*$. The q-SDH assumption suggests that any probabilistic polynomial-time (PPT) algorithm solving the q-SDH problem has negligible success probability. When q is any polynomial-time function on the security parameter we will write simply SDH.

The *Strong-RSA* problem [4] is as follows: given $n, z \in QR(n)$, where $QR(n)$ is the group of quadratic residues of $\mathbb{Z}_n^*$ asks for two integers $u, e > 1$ so that $u^e \equiv_n z$. The Strong-RSA assumption suggests that any PPT algorithm solving the Strong-RSA problem has negligible success probability.

The *Linear Decisional Diffie Hellman* assumption (Linear-DDH) [8] is as follows: the distribution of tuples of the form $(u, v, h, u^{\alpha}, v^{\beta}, h^{\alpha+\beta})$ where $u, v, h \leftarrow_R \mathbb{G}_1$ and $\alpha, \beta \leftarrow_R \mathbb{Z}_p$, is computationally indistinguishable from the distribution of tuples of the form $(u, v, h, u^{\alpha}, v^{\beta}, \eta)$ where where $u, v, h, \eta \leftarrow_R \mathbb{G}_1$ and $\alpha, \beta \leftarrow_R \mathbb{Z}_p$. The Linear-DDH is assumed to be true, even in groups where DDH fails (e.g., groups $\mathbb{G}_1$ for which we have a bilinear mapping).

The *Decisional Composite Residuosity* (DCR) assumption [27] is defined as follows: it is computationally hard to distinguish between the distributions of tuples of the form $(N, u^N \bmod N2)$ where N is an RSA safe composite modulus and $u \leftarrow_R \mathbb{Z}_{N2}^*$ and the distribution of tuples of the form (N, v) where N is an RSA safe composite modulus and $v \leftarrow_R \mathbb{Z}_N^*$.

3 Group Signatures with Concurrent Join: Modeling

In this section we give the formal definition of group signatures with concurrent join. First we start with the syntax of the signature. The parties that are involved in the scheme include the Group Manager (GM), the Users and the Verifiers.

Definition 1. *A group signature scheme with concurrent joins is a digital signature scheme that is comprised of the following five procedures:*

SETUP*: it is a probabilistic algorithm that on input a security parameter* 1^{ν}*, it outputs the group public key* $\mathcal{Y}$ *(including all system parameters) and the secret key* $\mathcal{S}$ *for the GM.* SETUP *initializes a public state string* $St = (St_{users}, St_{join-trans})$ *with two components* $St_{users} = \epsilon$ *and* $St_{join-trans} = \epsilon$*. The public state string* St *will hold the user identity database and the database of the Join protocol transcripts. This information will be publicly available and will grow as more users are introduced into the system.*

JOIN*: A* protocol *between the GM and a user that results in the user becoming a new group member. The user's output is a* membership certificate *and a* membership secret*. We will denote the i-th user's membership certificate by* cert_i *and the corresponding membership secret by* sec_i*.*
Since JOIN *is a two-party protocol, its specification includes the description of two interactive Turing Machines (ITM)* $\mathsf{J}_{\mathsf{user}}, \mathsf{J}_{\mathsf{GM}}$*. Only* $\mathsf{J}_{\mathsf{user}}$ *will have a private output.*

According to the notations of section 2 an execution of the protocol is denoted as $[\mathsf{J}_{\mathsf{user}}(St,\mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St,\mathcal{Y},\mathcal{S})]$ *and has two "output" components:*

1. *the user private output,* $\langle i, \mathsf{cert}_i, \mathsf{sec}_i\rangle \leftarrow \mathsf{User}[\mathsf{J}_{\mathsf{user}}(St,\mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St,\mathcal{Y},\mathcal{S})]$, *and*
2. *the public transcript,* $\mathsf{transcript}_i \leftarrow \mathsf{Trans}[\ \mathsf{J}_{\mathsf{user}}(St,\mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St,\mathcal{Y},\mathcal{S})]$.

After a successful execution of JOIN *the following updates are made:* $St_{users} = St_{users}||\langle i\rangle$ *and* $St_{join-trans} = St_{join-trans}||\ \langle i, \mathsf{transcript}_i\rangle$. *The identity-tag* i *will be selected from a space of possible identity tags denoted by* ID.

SIGN*: A probabilistic algorithm that given the group's public-key, a membership certificate, a membership secret and a message* m*, it outputs a group signature for the message* m*. We write* $\mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}_i, \mathsf{sec}_i, m)$ *to denote the application of the signing algorithm on the message* m.

VERIFY*: An algorithm for establishing the validity of an alleged group signature on a message with respect to a group public-key. If* σ *is a signature on a message* m*, then we have* $\mathtt{VERIFY}(\mathcal{Y}, m, \sigma) \in \{\top, \bot\}$.

OPEN*: An algorithm that, given a message, a valid group signature on it, a group public-key, the GM's secret-key and the public-state it determines the identity of the signer. In particular* $\mathtt{OPEN}(m, \sigma, St, \mathcal{Y}, \mathcal{S}) \in St_{users} \cup \{\bot\}$.

Notation. Below we will introduce a helpful notation for describing the relationship between transcripts and membership certificates and secrets. Given $\langle \mathcal{Y}, \mathcal{S}\rangle \leftarrow \mathtt{SETUP}(1^\nu)$ we define the following relations over strings based on $\mathcal{Y}$ and some public state St,

$\langle i, \mathsf{cert}, \mathsf{sec}\rangle \leftrightharpoons_{(\mathcal{Y},St)} \mathsf{transcript}$ if there exist coin tosses ρ for J_{GM} and $\mathsf{J}_{\mathsf{User}}$ so that

$$\langle i, \mathsf{cert}, \mathsf{sec}\rangle = \mathsf{User}[\mathsf{J}_{\mathsf{User}}(St,\mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St,\mathcal{Y},\mathcal{S})](\rho)$$

and

$$\mathsf{transcript} = \mathsf{Trans}[\mathsf{J}_{\mathsf{User}}(St,\mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St,\mathcal{Y},\mathcal{S})](\rho)$$

Similarly we will define $\mathsf{cert} \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}$, if there exist coin tosses ρ for J_{GM} and $\mathsf{J}_{\mathsf{User}}$ and a state St so that

$$\langle i, \mathsf{cert}, \mathsf{sec}\rangle = \mathsf{User}[\mathsf{J}_{\mathsf{User}}(St,\mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St,\mathcal{Y},\mathcal{S})](\rho)$$

Finally we define the set of all valid public states Valid as follows: $St_0 \in \mathsf{Valid}$ if there exists a PPT Turing machine M and $\langle \mathcal{Y}, \mathcal{S}\rangle \leftarrow \mathtt{SETUP}(1^\nu)$ so that when $M^{\mathcal{I}_{St}[\mathtt{JOIN} \leftrightarrow GM(St,\mathcal{Y},\mathcal{S}), \mathsf{READ}_{St}]}$ terminates it holds that $St = St_0$ and the interface oracle $\mathcal{I}$ given to M initializes $St = (\epsilon, \epsilon)$ and allows M to have read access to St through READ queries (that $\mathcal{I}_{St}$ allowes to M in addition to $\mathtt{Start-Session}$ and $\mathtt{Advance-Session}$ queries). If $\mathcal{I}_{St}$ initializes St to some $St_0 \in \mathsf{Valid}$ that is not (ϵ, ϵ) then this defines the set of all *valid extensions* of the public-state St_0 that will be denoted by Valid_{St_0}. Obviously $\mathsf{Valid} = \mathsf{Valid}_{(\epsilon,\epsilon)}$.

Correctness. Below we define the correctness of a group signature scheme that satisfies the above syntax. Note that a group signature is a tuple $\langle \mathtt{SETUP}, \mathtt{JOIN}, \mathtt{SIGN}, \mathtt{VERIFY}, \mathtt{OPEN}\rangle$ with $\mathtt{JOIN} = \langle \mathsf{J}_{\mathsf{User}}, \mathsf{J}_{\mathsf{GM}}\rangle$.

Definition 2. *A group signature with concurrent join is correct if the following are true:*

C1. *(users are assigned unique names) For any* $St \in \mathsf{Valid}$ *it holds that* St_{users} *contains no multiply defined identity-tags, i.e., if* $St_{users} = \langle i_1 \rangle || \ldots || \langle i_K \rangle$ *it holds that* $j \neq j' \Rightarrow i_j \neq i_{j'}$.

C2. *(signing is correct) For any* $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^\nu)$*, any strings* $\mathsf{cert} \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}$ *and any* $m \in \{0,1\}^*$*, it holds that* $\mathtt{VERIFY}(\mathcal{Y}, m, \mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}, \mathsf{sec}, m)) = \top$.

C3. *(open is correct) For any* $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^\nu)$*, any* $St \in \mathsf{Valid}$*, any* $m \in \{0,1\}^*$*, and any* $\langle i, \mathsf{cert}, \mathsf{sec} \rangle \leftrightharpoons_{(\mathcal{Y},St)} \mathsf{transcript}$ *it holds that* $\mathtt{OPEN}(m, \mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}, \mathsf{sec}, m), St'', \mathcal{Y}, \mathcal{S}) = i$*, where* $St'' \in \mathsf{Valid}_{St'}$ *and* St' *is defined as follows:* $St'_{users} = St_{users} || \langle i \rangle$ *and* $St'_{join-trans} = St_{join-trans} || \langle i, \mathsf{transcript} \rangle$.

Property *C1* requires that the JOIN protocol assigns a different identity tag to all users. Property *C2* ensures the correctness of the underlying signing and verification for any valid signing key (that includes a membership secret and a membership certificate). Finally, property *C3* ensures that the OPEN algorithm correctly identifies all signers: in particular it says that if a user is introduced at some moment in the system's operation and the public-state St is updated with the user's identity tag resulting to state St' then it holds that whenever this user issues a group signature the user will be correctly identified for every public state St'' that succeeds the public-state St' of the system. We note that it may be viable to collapse *C1* and *C3* but, given the intuitiveness of the formulation, we keep them as separate properties.

Definition of Security. Security against group signatures with concurrent join, will be broken into three basic properties following the model designs of [23, 25]. The properties are formalized as games between the adversary and an entity called the interface, denoted by $\mathcal{I}$ that represents the *uncorrupted aspect of the system* in each attack.

Misidentification. In a misidentification attack, the adversary joins the system through possibly many concurrent sessions of the JOIN protocol and it attempts to produce a signature that cannot be opened to any of the users that are adversarially controlled. We note that without loss of generality we will assume that *all* users introduced in the system are adversarially controlled; this means that the goal of the adversary is to simply make the OPEN algorithm to fail. We remark that adversaries that make the OPEN algorithm to point to an innocent user will be handled in the framing attack (next paragraph).

Below, $\mathcal{I}_{St}[\mathtt{JOIN}_{\leftrightarrow GM}]$ will denote the interface oracle for concurrent simulation of the GM party in the protocol JOIN (refer to section 2 for the definition). Note that the interface $\mathcal{I}$ has access to the public state string St and it updates it accordingly whenever a new user (the adversary that is) successfully completes the JOIN dialog. Also, an oracle $\mathtt{READ}_{St}$ is provided to the adversary that allows him to read the contents of the public state database that contains the identification transcripts and user identity tags. Finally, an oracle OPEN is provided to the adversary that allows him to submit signatures and obtain the output of the opening algorithm.

The Misidentification-Attack Game $G^{\mathcal{A}}_{\mathsf{mis}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{mis}}(1^\nu)$):
1. $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^\nu)$; $St = (St_{users}, St_{join-trans}) = (\epsilon, \epsilon)$;
2. $\langle m, \sigma \rangle \leftarrow \mathcal{A}^{\mathcal{I}_{St}[\mathtt{JOIN}\leftrightarrow GM(St,\mathcal{Y},\mathcal{S}),\mathtt{READ}_{St},\mathtt{OPEN}]}(\mathcal{Y})$;
3. If $(\mathtt{VERIFY}(\mathcal{Y}, m, \sigma) = \top) \wedge (\mathtt{OPEN}(m, \sigma, \mathcal{Y}, \mathcal{S}, St) = \bot)$ then return $\top$ else $\bot$;

We will say that a group signature is secure against misidentification attacks with concurrent join provided that for all $\mathcal{A}$ it holds that $\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{mis}}(1^\nu) = \top] = 1 - \mathsf{negl}(\nu)$.

Framing. In a framing attack the adversary plays the role of the system where the interface represents a handful of innocent users. A framing attack is meant to capture any adversarial behavior that allows the adversary to make the open algorithm point to an innocent user. We remark that this captures the notion of exculpability as well as any other adversarial behavior that frames an innocent user. In the concurrent setting, we allow the adversary to initiate many concurrent executions of the JOIN dialog playing the role of a malicious GM. The goal of the adversary now is to produce a signature that opens to one of the innocent users.

Naturally in modeling such an attack we cannot allow to the adversary to do all the bookkeeping for the user database himself (otherwise an OPEN operation would be without meaning). Every time the adversary successfully terminates a JOIN dialog with an innocent user that is controlled by the interface $\mathcal{I}$, the interface will add the user identity into the St_{users} and will append the whole communication transcript to $St_{join-trans}$. Moreover it will keep a private database containing the secrets of the innocent users that will have the format $\langle i, \mathsf{sec}_i \rangle$ (these will not be accessible to the adversary). In addition to the above, we will allow the adversary to submit queries to a SIGN oracle that will be handled by the interface oracle $\mathcal{I}$ and accepts the identity of one of the innocent users and a message and returns a signature of this message with the signing mechanism of the named user.

We allow the adversary to have appropriately restricted modify access to the public-state St; this access will be handled by $\mathcal{I}$ in the form of the $\mathtt{MODIFY}_{St}$ oracle query. As mentioned already we will not give to the adversary full write capability to the public state St since if he is allowed to this, opening any signature correctly would be meaningless (e.g., if the adversary erases the database of JOIN transcripts it is straightforward that the opening capability is cancelled). The restrictions are as follows: $\mathtt{MODIFY}_{St}$ will not permit the adversary to insert a join transcript that reuses an identity tag (this restriction is essential to maintain the semantics of the OPEN unambiguous) and will not allow the adversary to modify any of the identity tags or join transcripts of the innocent users (to these the adversary will have read-only access). Any other modification of the public-state will be allowed by $\mathcal{I}$ (in particular the adversary is allowed to introduce users to the public-state as well as erase them — for this reason there is no need for a "corrupt" oracle).

We will use the notation $St^{\mathcal{I}}_{users}$ to denote all innocent users in the system that are introduced by the execution of the concurrent JOIN oracle and are managed by the interface oracle $\mathcal{I}$.

The Framing-Attack Game $G^{\mathcal{A}}_{\mathsf{fra}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{fra}}(1^\nu)$):
1. $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^\nu)$; $St = (St_{users}, St_{join-trans}) = (\epsilon, \epsilon)$;
2. $\langle m, \sigma \rangle \leftarrow \mathcal{A}^{\mathcal{I}[\mathtt{JOIN}_{\leftrightarrow User(\mathcal{Y})}, \mathtt{SIGN}, \mathtt{READ}_{St}, \mathtt{MODIFY}_{St}]}(\mathcal{Y}, \mathcal{S})$
3. $i = \mathtt{OPEN}(m, \sigma, St, \mathcal{Y}, \mathcal{S})$;
4. If $(\mathtt{VERIFY}(\mathcal{Y}, m, \sigma) = \top) \wedge (i \in St^{\mathcal{I}}_{users})$ then return $\top$ else return $\bot$;

We say that a group signature satisfies security against framing attacks with concurrent join provided that for all $\mathcal{A}$ it holds that $\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{fra}}(1^\nu) = \top] = 1 - \mathsf{negl}(\nu)$.

Anonymity. Finally, anonymity is modeled as a sort of CCA2 attack against the identity encryption embedding mechanism of the group signature.

The Anonymity-attack Game $G^{\mathcal{A}}_{\mathsf{anon}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{anon}}(1^\nu)$):
1. $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^\nu)$; $St = (St_{users}, St_{join-trans}) = (\epsilon, \epsilon)$;
2. $\langle aux, m, \mathsf{cert}_1, \mathsf{sec}_1, \mathsf{cert}_2, \mathsf{sec}_2, \rangle \leftarrow \mathcal{A}^{\mathcal{I}[\mathtt{JOIN}_{\leftrightarrow GM(St,\mathcal{Y},\mathcal{S})}, \mathtt{READ}_{St}, \mathtt{OPEN}]}(\mathsf{play}, \mathcal{Y})$
3. if $\neg((\mathsf{cert}_1 \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}_1) \wedge (\mathsf{cert}_2 \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}_2))$ or $\mathsf{cert}_1 = \mathsf{cert}_2$ then stop; return $\bot$;
4. Choose $b \leftarrow_R \{1, 2\}$;
5. $\psi \leftarrow \mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}_b, \mathsf{sec}_b, m)$;
6. $b^* \leftarrow \mathcal{A}^{\mathcal{I}[\mathtt{JOIN}_{\leftrightarrow GM(St,\mathcal{Y},\mathcal{S})}, \mathtt{READ}_{St}, \mathtt{OPEN}^{\neg\psi}]}(\mathsf{guess}, aux)$;
7. if $b = b^*$ return $\top$ else return $\bot$;

We note that the $\mathtt{OPEN}^{\neg\psi}$ oracle operates as the $\mathtt{OPEN}$ oracle with the usual restriction that it should return $\bot$ if the adversary submits ψ as the signature to be opened.

A group signature is said to be secure against anonymity attacks with concurrent join provided that for all $\mathcal{A}$ it holds that $2\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{anon}}(1^\nu) = \top] - 1 = \mathsf{negl}(\nu)$.

Based on all the above we will say that a group signature with concurrent join is *secure* provided that it is secure against misidentification, framing and anonymity attacks.

4 Group Signatures with Efficient Concurrent Join: Construction

In this section we describe our efficient group signature construction. A number of primitives proved to be instrumental in our construction, namely: BB signatures [7], Linear ElGamal encryption [8], and a CCA2 variant of Pailier encryption [27, 22, 15]. We first begin by describing the public-parameters our system will employ.

Public-parameters. The public parameters of the scheme are as follows:

- p1 two groups of order p where p is a ℓ_p-bit prime, $p > 2^{\ell_p - 1}$, denoted by $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$, so that there is e and $\mathbb{G}_T$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map.
- p2 an RSA-modulus n, of ℓ_n bits; n is selected so that Strong-RSA will be infeasible over $QR(n)$.

p3 three integer ranges S, S', S''. We define the integer range $S =_{\mathsf{df}} S(2^{\ell-1}, 2^{\ell_p-2}) = \{2^{\ell-1}-2^{\ell_p-2}+1, \ldots, 2^{\ell-1}+2^{\ell_p-2}-1\}$. Observe that if $x, y \in S(2^{\ell-1}, 2^{\ell_p-2})$ and $x \equiv_p y$ then it holds that $x = y$; indeed, $p \mid x - y$ means that $x = y + kp$; assume without loss of generality that $k \geq 0$. Now, since $2^{\ell_p-1} < p < 2^{\ell_p}$ we have that $y \geq x + 2^{\ell_p-1}$; this is a contradiction, since even if $x = \min S = 2^{\ell-1} - 2^{\ell_p-2} + 1$ we have that $y \geq 2^{\ell-1} + 2^{\ell_p-1} - 2^{\ell_p-2} + 1 \geq 2^{\ell-1} + 2^{\ell_p-2} + 1 > \max S$. It follows that $k = 0$ and as a result $x = y$.
Now let $k, \epsilon > 1$ be parameters and select the ranges S', S'' as follows: $S' =_{\mathsf{df}} S(2^{\ell'}, 2^{\mu'}) = S(2^{\ell-1}, 2^{\lfloor(\ell_p-4)/\epsilon\rfloor-k})$ and $S'' =_{\mathsf{df}} S(2^{\ell''}, 2^{\mu''}) = S(2^{\ell'/2}, 2^{\mu'/2})$, so that $S(2^{\ell''}, 2^{\epsilon\mu''+k+2})$ does not contain an integer smaller than 2 and is disjoint from the range S. The ranges S, S', S'' are assumed subsets of $\{1, \ldots, \phi(n)\}$.

p4 a safe RSA-modulus N of ℓ_N bits with $N = PQ$ and $P = 2P'+1, Q = 2Q'+1$, so that in the group $\mathbb{Z}^*_{N2}$ it holds that the DCR assumption is hard, and the value $G = (G_0)^{2N} (\mathrm{mod} N2)$ is selected with $G_0 \leftarrow_R \mathbb{Z}^*_{N2}$. Note that with overwhelming probability $\langle G \rangle$ is the subgroup of quadratic residues modulo $N2$ that are simultaneously N-th residues; note that $\#\langle G \rangle = P'Q'$.

Regarding the size of parameters we observe the following: ℓ_p can be quite small, e.g., 170 bits is sufficient to achieve security that is equivalent to security of 1020 bits in multiplicative groups for the discrete-log problem (cf. also [8]). On the other hand ℓ_n, ℓ, ℓ_N will be selected so that an RSA modulus with this number of bits is hard to factor and thus $\ell_n, \ell, \ell_N \geq 1024 \gg \ell_p$. The above public parameters will be selected by the setup procedure of the group signature system as described below.

SETUP. The procedure first generates the public-parameters p1 and p2 and p3 as described above. Then, it executes the following steps:

- It selects two values $\gamma, \delta \leftarrow_R \mathbb{Z}_p$ and sets $w = g_2^{\gamma}$ and $v = g_2^{\delta}$; this is the setup for BB signatures, cf. [7].
- It selects two values $\alpha, \beta \leftarrow_R \mathbb{Z}_p$ and $u \leftarrow_R \mathbb{G}_1$ and sets $u' = u^{\alpha/\beta}$ and $h = u^{\alpha}(u')^{\beta}$; observe that it holds that $u^{\alpha} = (u')^{\beta}$. This is the key-setup for linear ElGamal encryption, cf. [8].
- It selects $g, f_1, f_2, f_3 \leftarrow_R QR(n)$. These will be used for commitments.
- (Opening functionality) the public parameters N, G according to p4 are selected as well as $H_1, H_2, H_3 \in \langle G \rangle$ with $H_i = G^{a_i}$, $a_i \leftarrow_R \mathbb{Z}_{\lfloor N/4 \rfloor}$ for $i = 1, 2, 3$ and a hash-key hk for a universal one-way hash function family UOHF. We remark that this step can be entirely separated from the GM's setup phase and executed by an opening authority. Nevertheless for convenience and simplification of the presentation we do not make further distinction in the present version of the paper.

The public-key $\mathcal{Y}$ is set to $\langle g_1, g_2, u, u', h, w, v, \mathrm{desc}(\mathbb{G}_1||\mathbb{G}_2||\mathbb{G}_T||e||\mathsf{UOHF}), g, f, n, N, G, H_1, H_2, H_3, \mathsf{hk}\rangle$ and the secret key $\mathcal{S}$ is set to to $\langle \gamma, \delta; a_1, a_2, a_3\rangle$. Note that the factorization of n as well as the values α, β (the decryption key for the linear ElGamal encryption) are not needed and thus they are discarded.

JOIN. In the join protocol execution, the user will obtain a BB signature on an RSA modulus that he selects. A user's membership certificate is the signature itself together with the RSA modulus; a user's membership secret on the other hand is the factorization of the modulus. The join procedure between a prospective user and the GM is described in detail below:

- (User→GM) The user initiates the procedure and selects random $x \in S'$ to be an ℓ-bit RSA modulus with x_1, x_2 its two prime divisors, so that $x_1 \in S''$. The User transmits x.
- (GM→User) The GM checks whether $x \in S'$ and whether x was submitted by another user in a previous JOIN; if the check fails the GM terminates the JOIN protocol; otherwise (i) it reads the public-state St, selects $i \in \mathsf{ID}$ so that $i \notin St_{users}$ and in such a manner that i is distinct from any other concurrent executions and writes to its communication tape the values $\langle i, \sigma, r\rangle$ where $r \leftarrow_R \mathbb{Z}_p$ and $\sigma = g_1^{1/(\gamma+x+\delta r)}$; finally it updates $St_{join-trans}$ by appending to it the tuple $\langle i, \sigma, r\rangle$ and sets $St_{users} = St_{users}||\langle i\rangle$.
- The user verifies that $e(\sigma, wg_2^x v^r) = e(g_1, g_2)$ and that $i \notin St_{users}$; if either test fails the user fails the JOIN dialog. Otherwise, it terminates successfully by setting his membership certificate to $\mathsf{cert} = \langle x, \sigma, r\rangle$ and his membership secret to $\mathsf{sec} = \langle x_1, x_2\rangle$.

Observe that the user *does not prove* that x was selected appropriately; Perhaps surprisingly, we show that this is still sufficient for security in the concurrent setting. Naturally if the user chooses x inappropriately two things may happen: (i) the user may not be able to issue group signatures, e.g., this may happen when x is a prime; this naturally is of no concern to the GM, (ii) the user selects x as an integer that is easy to factor; while this is of concern there is nothing that can be done about it: this case is conceptually the same as the case that the user just leaks its secret-key; while this possibility is annoying there is little that can be done to prevent this in any group signature scheme.

As a side-note the reader perhaps would be concerned with the fact that the BB-signature above (that typically operates over short messages of, say, about 170 bits) will be used to sign RSA moduli that are over 1000 bits. Our scheme prevents any kind of naive forgery based on the wrap-around by employing range proofs that ensure that the integers employed by users, while they are large they all fall within an integer range S' that contains sufficiently less than 2^{170} elements (cf. the parameter selection p3).

SIGN. We present the signing algorithm. The user possesses the following: a membership certificate $\langle x, \sigma, r\rangle$ and the corresponding membership secret x_1, x_2. The signing algorithm will be obtained by applying the Fiat-Shamir Heuristic on an appropriately designed proof of knowledge. First, the signer computes the following values:

$T_1 = u^z$	$z \leftarrow_R \mathbb{Z}_p$ in $\mathbb{G}_1$
$T_2 = (u')^{z'}$	$z' \leftarrow_R \mathbb{Z}_p$ in $\mathbb{G}_1$
$T_3 = h^{z+z'}\sigma$	in $\mathbb{G}_1$
$T_4 = g^y f_1^{x_1}$	$y \leftarrow_R S(1, 2^{\ell_n - 2})$ in $QR(n)$
$T_5 = g^{y'} f_2^{x_2} f_3^t$	$y' \leftarrow_R S(1, 2^{\ell_n - 2})$ in $QR(n)$
$C_0 = G^t$	$t \leftarrow_R S(1, 2^{\ell_N - 2})$ in $\mathbb{Z}^*_{N2}$
$C_1 = H_1^t (1+N)^x$	in $\mathbb{Z}^*_{N2}$
$C_2 = \|\|(H_2 H_3^{\mathcal{H}(\mathsf{hk}, C_0, C_1)})^t\|\|$	in $\mathbb{Z}^*_{N2}$

Note that $||x|| = x$ if $x \leq N2/2$ and $||x|| = N2 - x$ otherwise. Also recall that $S(a,b) =_{\mathsf{df}} \{a-b, \ldots, a+b\}$. Subsequently the signer will construct the signature "of knowledge" on the given message m by providing a proof of knowledge for the relations given in figure 2 that involve the fourteen witnesses $\theta_z, \theta_{z'}, \theta_x, \theta_{xz}, \theta_{xz'}, \theta_r, \theta_{rz}, \theta_{rz'}, \theta_{x_1}, \theta_{x_2}, \theta_y, \theta_{y'}, \theta_{yx_2}, \theta_t$.

$$T_1^{-1} u^{\theta_z} = 1 \qquad T_2^{-1} (u')^{\theta_{z'}} = 1$$
$$T_1^{-\theta_x} u^{\theta_{xz}} = 1 \qquad T_2^{-\theta_x} (u')^{\theta_{xz'}} = 1$$
$$T_1^{-\theta_r} u^{\theta_{rz}} = 1 \qquad T_2^{-\theta_r} (u')^{\theta_{rz'}} = 1$$
$$e(T_3, v)^{\theta_r} e(T_3, g_2)^{\theta_x} e(h, g_2)^{-\theta_{xz} - \theta_{xz'}} e(h, v)^{-\theta_{rz} - \theta_{rz'}} e(h, w)^{-\theta_z - \theta_{z'}} e(T_3, w) = e(g_1, g_2)$$
$$T_4^{-1} g^{\theta_y} f^{\theta_{x_1}} = 1 \qquad T_4^{-\theta_{x_2}} g^{\theta_{yx_2}} f^{\theta_x} = 1$$
$$T_5^{-1} g^{\theta_{y'}} f^{\theta_{x_2}} f_2^{\theta_t} = 1$$
$$C_0 = G^{\theta_t} \qquad C_1 = H_1^{\theta_t} (1+N)^{\theta_x} \qquad (C_2)2 = (H_2 H_3^{\mathcal{H}(\mathsf{hk}, C_0, C_1)})^{2\theta_t}$$
$$\theta_x \in S' \qquad \theta_{x_1} \in S''$$

Fig. 2. The relations defining the signature of knowledge

Given the coin tosses of the signer for the selection of $T_1, T_2, T_3, T_4, T_5, C_0, C_1$, the witnesses needed in figure 2 are selected as follows: $\theta_z = z, \theta_{z'} = z', \theta_x = x, \theta_{xz} = x \cdot z (\mathrm{mod} p)$, $\theta_{xz'} = x \cdot z' (\mathrm{mod} p)$, $\theta_r = r$, $\theta_{rz} = r \cdot z (\mathrm{mod} p), \theta_{rz'} = r \cdot z' (\mathrm{mod} p), \theta_{x_1} = x_1, \theta_{x_2} = x_2, \theta_y = y, \theta_{y'} = y', \theta_{yx_2} = y \cdot x_2$ in $\mathbb{Z}$, $\theta_t = t$. Now, given a message m, the signature will be constructed as follows:

1. (choose blindings) the values $\rho_z, \rho_{z'}, \rho_{xz}, \rho_{xz'}, \rho_r, \rho_{rz}, \rho_{rz'} \leftarrow_R \mathbb{Z}_p$ and $\rho_x \leftarrow_R \pm\{0,1\}^{\epsilon\mu' + k}$, $\rho_{x_1} \leftarrow_R \pm\{0,1\}^{\epsilon\mu'' + k}$ and $\rho_{x_2}, \rho_y, \rho_{y'} \leftarrow_R \pm\{0,1\}^{\epsilon(\ell_n - 2) + k}$, $\rho_{yx_2} \leftarrow_R \pm\{0,1\}^{2\epsilon(\ell_n - 2) + k}$ and $\rho_t \leftarrow_R \pm\{0,1\}^{\epsilon(\ell_N - 2) + k}$ are selected. Using these values the following are computed:

$$R_1 = u^{\rho_z} \qquad R_2 = (u')^{\rho_{z'}}$$
$$R_3 = T_1^{\rho_x} u^{-\rho_{xz}} \qquad R_4 = T_2^{\rho_x} (u')^{-\rho_{xz'}}$$
$$R_5 = T_1^{\rho_r} u^{-\rho_{rz}} \qquad R_6 = T_2^{\rho_r} (u')^{-\rho_{rz'}}$$
$$R_7 = e(T_3, v)^{\rho_r} e(T_3, g_2)^{\rho_x} e(h, g_2)^{-\rho_{xz} - \rho_{xz'}} e(h, v)^{-\rho_{rz} - \rho_{rz'}} e(h, w)^{\rho_z + \rho_{z'}}$$
$$R_8 = g^{\rho_y} f_1^{\rho_{x_1}} \qquad R_9 = T_4^{\rho_{x_2}} g^{-\rho_{yx_2}} f^{-\rho_x}$$

$$R_{10} = g^{\rho_{y'}} f_2^{\rho_{x_2}} f_3^{\rho_t}$$

$$R_{11} = G^{\rho_t} \qquad R_{12} = H_1^{\rho_t}(1+N)^{\rho_x} \qquad R_{13} = (H_2 H_3^{\mathcal{H}(\mathsf{hk},C_0,C_1)})^{2\rho_t}$$

2. (calculate challenge) using a hash function denoted by HASH the value

$$c \leftarrow \mathtt{HASH}(m||T_1||\ldots||T_4||T_5||R_1||\ldots||R_{12},R_{13})$$

is computed. The range of HASH is considered to be $\{0,1\}^k$.

3. (calculate response) Subsequently the following values are computed:

$s_z = \rho_z - cz$	in $\mathbb{Z}_p$	$s_{z'} = \rho_{z'} - cz'$	in $\mathbb{Z}_p$
$s_{xz} = \rho_{xz} - cxz'$	in $\mathbb{Z}_p$	$s_{xz'} = \rho_{xz'} - cxz'$	in $\mathbb{Z}_p$
$s_r = \rho_r - cr$	in $\mathbb{Z}_p$	$s_{rz'} = \rho_{rz'} - crz'$	in $\mathbb{Z}_p$
$s_{rz'} = \rho_{rz'} - crz'$	in $\mathbb{Z}_p$	$s_x = \rho_x - c(x - 2^{\ell'})$	in $\mathbb{Z}$
$s_{x_1} = \rho_{x_1} - c(x_1 - 2^{\ell''})$	in $\mathbb{Z}$	$s_{x_2} = \rho_{x_2} - c(x_2 - 1)$	in $\mathbb{Z}$
$s_y = \rho_y - c(y-1)$	in $\mathbb{Z}$	$s_{y'} = \rho_{y'} - c(y'-1)$	in $\mathbb{Z}$
$s_{yx_2} = \rho_{yx_2} - c(y \cdot x_2 - 2)$	in $\mathbb{Z}$	$s_t = \rho_t - c(t-1)$	in $\mathbb{Z}$

The output of the signing algorithm is the tuple: $\langle T_1, T_2, T_3, T_4, T_5, C_0, C_1, C_2, c, s_z, s_{z'}, s_{xz}, s_{xz'}, s_r, s_{rz}, s_{rz'}, s_x, s_{x_1}, s_{x_2}, s_y, s_{y'}, s_{yx_2}, s_t\rangle$.

VERIFY. Signature verification is achieved by the following tests:

$$s_x \stackrel{?}{\in} \pm\{0,1\}^{\epsilon\mu'+k+1} \wedge s_{x_1} \stackrel{?}{\in} \pm\{0,1\}^{\epsilon\mu''+k+1} \wedge C_0, C_1, C_2 \stackrel{?}{\in} \mathbb{Z}^*_{N2} \wedge C_2 \stackrel{?}{\leq} N2/2$$

$$\begin{aligned} c \stackrel{?}{=} \mathsf{HASH}\Big(& m \,||\, T_1 \,||\, T_2 \,||\, T_3 \,||\, T_4 \,||\, T_5 \,|| \\ & || \, u^{s_z} T_1^c || (u')^{s_{z'}} T_2^c \\ & || \, T_1^{-s_x + c2^{\ell'}} u^{s_{xz}} || T_2^{-s_x + c2^{\ell'}} (u')^{s_{xz'}} \\ & || \, T_1^{s_r} u^{-s_{rz}} || T_2^{s_r} (u')^{-s_{rz'}} \\ & || \, e(T_3, v)^{s_r} e(T_3, g_2)^{s_x - c2^{\ell'}} e(h, g_2)^{-s_{xz} - s_{xz'}} \\ & \quad e(h, v)^{-s_{rz} - s_{rz'}} e(h, w)^{s_z + s_{z'}} e(g_1, g_2)^c e(T_3, w)^{-c} \\ & || \, T_4^c g^{s_y - c} f_1^{s_{x_1} - c2^{\ell''}} || T_4^{s_{x_2} - c} g^{-s_{yx_2} + 2c} f_1^{-s_x + c2^{\ell'}} || T_5^c g^{s_{y'} - c} f_2^{s_{x_2} - c} f_3^{s_t - c} \\ & || C_0^c G^{s_t - c} || C_1^c H_1^{s_t - c} (1+N)^{s_x - c2^{\ell'}} || C_2^c (H_2 2 H_3^{2\mathcal{H}(\mathsf{hk},C_0,C_1)})^{s_t - c} \Big) \end{aligned}$$

OPEN. Given a signature as described above: first the signature is verified as well as the relation $(C_2)2 = C_0^{2(a_2 + \theta\mathcal{H}(\mathsf{hk},C_0,C_1)}$ is checked. if the test passes. If any check fails OPEN returns $\perp$. Otherwise, OPEN computes $\tilde{m} = C_1 C_0^{-a_1}$; due to the properties enforced by the proof of knowledge (cf. figure 2) it holds that $x = (\tilde{m} - 1)/N$. Then, the OPEN algorithm searches $St_{join-trans}$ for transcripts of the form $\langle j, x_j, \sigma_j, r_j\rangle$ with $x_j = x$ the identity j of the signer is recovered. If no such x_j is found, OPEN returns $\perp$.

5 Proof of Security

The proof of security is described here, it relies on the random oracle model (we prove the group signature rather than the interactive identity escrow scheme).

Theorem 1. *The signature of knowledge that specifies the* `SIGN` *algorithm satisfies: completeness, special soundness under the Strong-RSA assumption and statistical honest verifier zero-knowledge.*

Theorem 2. *Any misidentification attacker in the random oracle model against our group signature can be transformed to an adaptive chosen message attacker in the standard model against the BB signature assuming the Strong-RSA assumption.*

Theorem 3. *Any framing attacker in the random oracle model against our group signature can be transformed to a factoring algorithm in the standard model assuming the Strong-RSA assumption.*

Theorem 4. *Any anonymity adversary against our group signature in the random oracle model can be transformed to a CCA2 attacker against the public-key encryption that is employed in our scheme; this is conditional on the validity of (i) the Linear-DDH assumption, (ii) the assumption that the digital signature scheme employed (BB-signature) satisfies strong existential unforgeability. (iii) the DLOG assumption over the subgroup of $2N$-th residues inside $\mathbb{Z}^*_{N^2}$.*

The above three theorems culminate to the following theorem:

Theorem 5. *Our group signature is correct and secure in the random oracle model under the assumptions: SDH, Linear-DDH, Strong-RSA and DCR assumptions.*

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Advances in Cryptology – CRYPTO ' 2000*, volume 1880 of *Lecture Notes in Computer Science*. International Association for Cryptologic Research, Springer, 2000.
2. G. Ateniese and B. de Medeiros. Efficient group signatures without trapdoors. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, Lecture Notes in Computer Science. International Association for Cryptologic Research, Springer, 2003.
3. G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In M. Franklin, editor, *Financial cryptography: Third International Conference, FC '99, Anguilla, British West Indies, February 22–25, 1999: proceedings*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1999.

4. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1997.
5. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, Warsaw, Poland, 2003. Springer.
6. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. Cryptology ePrint Archive, Report 2004/077, 2004. `http://eprint.iacr.org/`.
7. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT ' 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, Interlaken, Switzerland, 2004. Springer.
8. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
9. J. Camenisch. Efficient and generalized group signatures. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques*, Lecture Notes in Computer Science, pages 465–479. International Association for Cryptologic Research, Springer, 1997.
10. J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks - SCN 2004*. Springer, 2003.
11. J. Camenisch and A. Lysyanskaya. An identity escrow scheme with appointed verifiers. In J. Kilian, editor, *Advances in Cryptology – CRYPTO ' 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 388–407. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2001.
12. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*. Springer, 2004.
13. J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In K. Ohta and D. Pei, editors, *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, volume 1514 of *Lecture Notes in Computer Science*, pages 160–174. International Association for Cryptologic Research, Springer-Verlag, 1998.
14. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes (extended abstract). In M. j. Wiener, editor, *19th International Advances in Cryptology Conference – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 413–430. Springer, 1999.
15. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*. Springer-Verlag, 2003.
16. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. S. K. Jr., editor, *Advances in Cryptology – CRYPTO ' 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. International Association for Cryptologic Research, Springer, 1997.
17. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology, EUROCRYPT 1991 (Lecture Notes in Computer Science 547)*, pages 257–265. Springer-Verlag, April 1991. Brighton, U.K.

18. L. Chen and T. P. Pedersen. New group signature schemes (extended abstract). In A. D. Santis, editor, *Advances in Cryptology—EUROCRYPT 94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1995, 9–12 May 1994.
19. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT ' 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626, Interlaken, Switzerland, 2004. Springer.
20. P.-A. Fouque and J. Stern. One round threshold discrete-log key generation without private channels. In K. Kim, editor, *Public Key Cryptography — 4th International Workshop on Practice and Theory in Public Key Cryptosystems*, volume 1992 of *Lecture Notes in Computer Science*, pages 300–316, Cheju Island, Korea, 2001. Springer.
21. J. Garay, P. D. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 177–194, Warsaw, Poland, 2003. Springer.
22. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, Warsaw, Poland, 2003. Springer.
23. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT ' 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589, Interlaken, Switzerland, 2004. Springer.
24. A. Kiayias and M. Yung. Extracting group signatures from traitor tracing schemes. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 630–648, Warsaw, Poland, 2003. Springer.
25. A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/`.
26. J. Kilian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 169–185. International Association for Cryptologic Research, Springer, 1998.
27. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, 1999.
28. G. Tsudik and S. Xu. Accumulating composites and improved group signing. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, Lecture Notes in Computer Science, pages 269–286. International Association for Cryptologic Research, Springer, 2003.

Floating-Point LLL Revisited

Phong Q. Nguyễn[1,⋆] and Damien Stehlé[2]

[1] CNRS/École normale supérieure, DI, 45 rue d'Ulm, 75005 Paris, France
Phong.Nguyen@di.ens.fr
http://www.di.ens.fr/~pnguyen/
[2] Univ. Nancy 1/LORIA, 615 rue du J. Botanique, 54602 Villers-lès-Nancy, France
Damien.Stehle@loria.fr
http://www.loria.fr/~stehle/

Abstract. The Lenstra-Lenstra-Lovász lattice basis reduction algorithm (LLL or L^3) is a very popular tool in public-key cryptanalysis and in many other fields. Given an integer d-dimensional lattice basis with vectors of norm less than B in an n-dimensional space, L^3 outputs a so-called L^3-reduced basis in polynomial time $O(d^5 n \log^3 B)$, using arithmetic operations on integers of bit-length $O(d \log B)$. This worst-case complexity is problematic for lattices arising in cryptanalysis where d or/and $\log B$ are often large. As a result, the original L^3 is almost never used in practice. Instead, one applies floating-point variants of L^3, where the long-integer arithmetic required by Gram-Schmidt orthogonalisation (central in L^3) is replaced by floating-point arithmetic. Unfortunately, this is known to be unstable in the worst-case: the usual floating-point L^3 is not even guaranteed to terminate, and the output basis may not be L^3-reduced at all. In this article, we introduce the L^2 algorithm, a new and natural floating-point variant of L^3 which provably outputs L^3-reduced bases in polynomial time $O(d^4 n(d + \log B) \log B)$. This is the first L^3 algorithm whose running time (without fast integer arithmetic) provably grows only quadratically with respect to $\log B$, like the well-known Euclidean and Gaussian algorithms, which it generalizes.

Keywords: LLL, L^3, Lattice Reduction, Public-Key Cryptanalysis.

1 Introduction

Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d$ be linearly independent vectors in $\mathbb{R}^n$ with $n \geq d$: often $n = d$ or $n = O(d)$. We denote by $L[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d] = \left\{ \sum_{i=1}^{d} x_i \boldsymbol{b}_i \mid x_i \in \mathbb{Z} \right\}$ the set of all integer linear combinations of the $\boldsymbol{b}_i$'s. This set is called a *lattice* and $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ a *basis* of that lattice. A lattice basis is usually not unique, but all the bases have the same number d of elements, called the *dimension* of the lattice. If

⋆ The work described in this article has in part been supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 215–233, 2005.

$d \geq 2$, there are infinitely many bases, but some are more interesting than others: they are called *reduced*. Roughly speaking, a reduced basis is a basis made of reasonably short vectors which are almost orthogonal. Finding good reduced bases has proved invaluable in many fields of computer science and mathematics (see [12, 8]), particularly in cryptology (see [30, 24]). This problem is known as *lattice reduction* and can intuitively be viewed as a vectorial generalisation of gcd computations.

The first breakthrough in lattice reduction dates back to 1981 with Lenstra's celebrated work on integer programming [20, 21], which was, among others, based on a novel lattice reduction technique (which can be found in the preliminary version [20] of [21]). Lenstra's reduction technique was only polynomial-time for fixed dimension, which was however sufficient in [20]. This inspired Lovász to develop a polynomial-time variant of the algorithm, which reached a final form in the seminal paper [19] where Lenstra, Lenstra and Lovász applied it to factor rational polynomials in polynomial time (back then, a famous problem), from which the name LLL or L^3 comes. Further refinements of the L^3 algorithm were later proposed, notably by Schnorr [33, 34]. Reduction algorithms (in particular L^3) have arguably become the most popular tool in public-key cryptanalysis (see the survey [30]). In the past twenty-five years, they have been used to break many public-key cryptosystems, including knapsack cryptosystems [31], RSA in particular settings [9, 7, 6], DSA and similar signatures in particular settings [14, 26], etc.

Given as input an integer d-dimensional lattice basis whose n-dimensional vectors have norm less than B, L^3 outputs a so-called L^3-reduced basis in time $O(d^5 n \log^3 B)$ without fast integer arithmetic, using arithmetic operations on integers of bit-length $O(d \log B)$. This worst-case complexity turns out to be problematic in practice, especially for lattices arising in cryptanalysis where d or/and $\log B$ are often large. For instance, in a typical RSA application of Coppersmith's lattice-based theorem [9], we may need to reduce a 64-dimensional lattice with vectors having RSA-type coefficients (1024-bit), in which case the complexity becomes "$d^5 n \log^3 B = 2^{66}$ ". As a result, the original L^3 algorithm is seldom used in practice. Instead, one applies floating-point (*fp*) variants, where the long-integer arithmetic required by Gram-Schmidt orthogonalisation (which plays a central role in L^3) is replaced by floating-point arithmetic (*fpa*) on much smaller numbers. The use of *fpa* in L^3 goes back to the early eighties when L^3 was used to solve low-density knapsacks [17]. Unfortunately, *fpa* may lead to stability problems, both in theory and practice, especially when the dimension increases: the running time of *fp* variants of L^3 such as Schnorr-Euchner's [36] is not guaranteed to be polynomial nor even finite, and the output basis may not be L^3-reduced at all. This phenomenon is well-known to L^3 practitioners, and is usually solved by sufficiently increasing the precision. For instance, experimental problems arose during the cryptanalyses [29, 25] of lattice-based cryptosystems, which led to improvements in Shoup's NTL library [41].

There is however one provable *fp*-variant of L^3, due to Schnorr [34], which significantly improves the worst-case complexity. Schnorr's variant outputs an

approximate L^3-reduced basis in time $O(d^3 n \log B(d + \log B)^2)$, using $O(d + \log B)$ precision *fp* numbers. However, this algorithm is mostly of theoretical interest and is not implemented in any of the main computational libraries [41, 22, 4, 1]. This can be explained by at least three reasons: it is not clear which *fpa*-model is used, the algorithm is difficult to describe, and the hidden complexity constants are rather large. More precisely, the required precision of *fp* numbers in [34] seems to be higher than $12d + 7\log_2 B$.

Our Results. We present the L^2 algorithm, a new and simple *fp*-variant of L^3 in a standard *fpa*-model, which provably outputs approximate L^3-reduced bases in polynomial time. More precisely, its complexity is $O(d^4 n(d + \log B)\log B)$ using only a $(d\log_2 3)$-bit precision, which is independent of $\log B$. This is the first L^3 whose running time grows only quadratically with respect to $\log B$ (hence the name L^2), whereas the growth is cubic – without fast integer arithmetic – for all other provable L^3 algorithms known. This improvement is significant for lattices where $\log B$ is larger than d, for example those arising from minimal polynomials [8] and Coppersmith's technique [9]. Interestingly, L^3 can be viewed as a generalisation of the famous Euclidean and Gaussian algorithms whose complexities are quadratic, not cubic like the original L^3. This arguably makes L^2 closer to Euclid's algorithm.

	L^3[19]	Schnorr[34]	L^2
Required precision	$O(d \log B)$	$> 12d + 7\log_2 B$	$d\log_2 3 \approx 1.58d$
Complexity	$O(d^5 n \log^3 B)$	$O(d^3 n(d + \log B)^2 \log B)$	$O(d^4 n(d + \log B)\log B)$

Fig. 1. Comparison of different L^3 algorithms

The L^2 algorithm is based on several improvements, both in the L^3 algorithm itself and more importantly in its analysis. From an algorithmic point of view, we improve the accuracy of the usual Gram-Schmidt computations by a systematic use of the Gram matrix, and we adapt Babai's nearest plane algorithm [3] to *fpa* in order to stabilize the so-called size-reduction process extensively used in L^3. We give tight bounds on the accuracy of Gram-Schmidt computations to prove the correctness of L^2. The analysis led to the discovery of surprisingly bad lattices: for instance, we found a 55-dimensional lattice [28] with 100-bit vectors which makes NTL's LLL_FP [41] (an improved version of [36]) loop forever, which contradicts [16] where it is claimed that double precision is sufficient in [36] to L^3-reduce lattices up to dimension 250 with classical Gram-Schmidt. However, for random looking lattice bases, stability problems seem to arise only in dimension much higher than 55, due perhaps to the well-known experimental fact that for such input bases, L^3 outputs better bases than for the worst-case. Finally, to establish a quadratic running time, we generalize a well-known cascade phenomenon in the complexity analysis of the Gaussian and Euclidean algorithms. This was inspired by the so-called greedy lattice reduction algorithm of [27], which is quadratic in low dimension thanks to another cascade. The cascade analysis is made possible by the efficiency of our *fp*-variant of Babai's

algorithm, and cannot be adapted to the standard L^3 algorithm. Besides, our tight bound on Babai's algorithm may be of independent interest. For instance, in Micciancio's variant [23] of the GGH cryptosystem [10], Babai's algorithm is used to decrypt.

Related Work. Much work [38, 34, 42, 15, 16, 35] has been devoted to improve L^3, specifically the exponent of d in the complexity, but none has improved the $\log^3 B$ factor (except [44, 39] for dimension two). Some of these improvements might be adaptable to L^2.

Floating-point stability has long been a mysterious issue in L^3. When it was realized during experiments that classical Gram-Schmidt orthogonalisation could be very unstable, it was suggested in the late nineties to use well-known alternative techniques (see [18, 11]) like Givens rotations (implemented in NTL) or Householder reflections, which are more expensive but seem to be more stable in practice. However, from a theoretical point of view, the best results known on the worst-case accuracy of such techniques are not significantly better than the so-called Modified Gram-Schmidt algorithm. Besides, most numerical analysis results refer to backward stability and not accuracy: such a mistake is made in [16], where a theorem from [18] is incorrectly applied. At the moment, it is therefore not clear how to exploit known results on Givens rotations and Householder reflections to improve L^3 theoretically. This is why L^2 only uses a process close to classical Gram-Schmidt.

Road map. In Section 2 we provide necessary background on lattices and L^3. We describe the L^2 algorithm in Section 3. Section 4 proves the correctness of L^2, while Section 5 analyzes its complexity. Additional information (such as complete proofs of technical lemmata and experimental results) will be given in the journal version of the present work.

Remarks. Like L^3, the L^2 algorithm works in fact with the underlying quadratic form and can therefore be used to reduce positive definite integer quadratic forms. It can be checked (see the full version) that L^2 can be extended to linearly dependent vectors, leading to what is to our knowledge the fastest algorithm known to construct a lattice basis from a generating set. Schnorr pointed out that the required precision in L^2 can be slightly decreased by using a better summation algorithm than ours (e.g. a tree-like algorithm), which may be interesting if one is restricted to a fixed precision. For the sake of simplicity, we keep a basic summation algorithm.

2 Background

Notation. All logarithms are in base 2. Let $\|.\|$ and $\langle ., . \rangle$ be the Euclidean norm and inner product of $\mathbb{R}^n$. The notation $\lceil x \rfloor$ denotes a closest integer to x. Bold variables are vectors. All the lattices we consider are integer lattices, as usual. The complexity model we use is the RAM model, and the computational cost is measured in elementary operations on bits, without fast integer arithmetic [40]. Our *fpa*-model is a smooth extension of the IEEE-754 standard [2], as provided

by NTL [41] and MPFR [32]. With an ℓ-bit working precision, a *fp*-number is of the form $x = \pm m_x \cdot 2^{e_x}$ where the mantissa $m_x \in [1/2, 1)$ is ℓ-bit long and the exponent e_x is an integer. We expect all four basic *fp*-operations to be correctly rounded: the returned value $\diamond(a \text{ op } b)$ for $\text{op} \in \{+, -, /, *\}$ is a closest *fp*-number to $(a \text{ op } b)$. In our complexity analysis, we do not consider the cost of the arithmetic on the exponents: it can be checked that the exponents are integers of length $O(\log(d + \log B))$, so that the cost is indeed negligible.
We recall basic notions from algorithmic geometry of numbers (see [24]).

Gram matrix. Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d$ be vectors. Their *Gram matrix* $G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$ is the $d \times d$ symmetric matrix $(\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle)_{1 \leq i,j \leq d}$ formed by all the inner products.

Lattice volume. A lattice L has infinitely many lattice bases when $\dim(L) \geq 2$. Any two bases are related to each other by a unimodular matrix (integral matrix of determinant ± 1), and therefore the determinant of the Gram matrix of a basis only depends on the lattice. The square root of this determinant is the *volume* $\text{vol}(L)$ (or *determinant*) of the lattice.

Gram-Schmidt orthogonalisation. Let $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ be linearly independent vectors. The *Gram-Schmidt orthogonalisation* (GSO) $[\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_d^*]$ is the orthogonal family defined recursively as follows: $\boldsymbol{b}_i^*$ is the component of $\boldsymbol{b}_i$ orthogonal to the linear span of $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1}$. We have $\boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \boldsymbol{b}_j^*$ where $\mu_{i,j} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle / \|\boldsymbol{b}_j^*\|^2$. For $i \leq d$ we let $\mu_{i,i} = 1$. The lattice L spanned by the $\boldsymbol{b}_i$'s satisfies $\text{vol}(L) = \prod_{i=1}^{d} \|\boldsymbol{b}_i^*\|$. The GSO family depends on the order of the vectors. If the $\boldsymbol{b}_i$'s are integer vectors, the $\boldsymbol{b}_i^*$'s and the $\mu_{i,j}$'s are rational.

QR factorisation. The GSO corresponds to the "R" part of the $Q \cdot R$ factorisation of the matrix representing the basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$, where Q is an orthogonal matrix ($Q \cdot Q^t = Q^t \cdot Q = Id$) and R is lower triangular. If $R = (r_{i,j})$, for any i we have $r_{i,i} = \|\boldsymbol{b}_i^*\|^2$ and for any $i \geq j$ we have $r_{i,j} = \mu_{i,j} \|\boldsymbol{b}_j^*\|^2$. In what follows, the *GSO family* denotes the $r_{i,j}$'s and $\mu_{i,j}$'s. Some information is redundant in rational arithmetic, but in the context of our *fp* calculations, it is useful to have all these variables to minimize the number of arithmetic operations and thus the precision loss.

Size-reduction. A basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ is *size-reduced* with factor $\eta \geq 1/2$ if its GSO family satisfies $|\mu_{i,j}| \leq \eta$ for all $1 \leq j < i \leq d$. The i-th vector $\boldsymbol{b}_i$ is *size-reduced* if $|\mu_{i,j}| \leq \eta$ for all $j < i$. Size-reduction usually refers to $\eta = 1/2$, but it is essential for L^2 to allow larger η.

L^3-reduction. A basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ is L^3-reduced with factor (δ, η) where $1/4 < \delta \leq 1$ and $1/2 \leq \eta < \sqrt{\delta}$ if the basis is size-reduced with factor η and if its GSO satisfies the $(d-1)$ Lovász conditions $(\delta - \mu_{\kappa,\kappa-1}^2) r_{\kappa-1,\kappa-1} \leq r_{\kappa,\kappa}$ (or equivalently $\delta \|\boldsymbol{b}_{\kappa-1}^*\|^2 \leq \|\boldsymbol{b}_\kappa^* + \mu_{\kappa,\kappa-1} \boldsymbol{b}_{\kappa-1}^*\|^2$), which implies that the GSO vectors never drop too much. Such bases have useful properties (see [19, 8, 24]), like providing approximations to the shortest vector problem and the closest vector problem. In particular, their first vector is relatively short: $\|\boldsymbol{b}_1\| \leq (\delta - \eta^2)^{-(d-1)/4} \text{vol}(L)^{1/d}$. L^3-reduction usually refers to the factor $(3/4, 1/2)$ initially chosen in [19]. But the closer δ and η are respectively to 1 and $1/2$, the shorter $\boldsymbol{b}_1$ should be. In practice,

Input: A basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ and $\delta \in (1/4, 1)$.
Output: An L^3-reduced basis with factor $(\delta, 1/2)$.
1. Compute the rational GSO, i.e., all the $\mu_{i,j}$'s and $r_{i,i}$'s.
2. κ:=2. While $\kappa \leq d$ do
3. Size-reduce $\boldsymbol{b}_\kappa$ using Babai's algorithm (Fig. 3), which updates the GSO.
4. κ':=κ. While $\kappa \geq 2$ and $\delta r_{\kappa-1,\kappa-1} \geq r_{\kappa',\kappa'} + \sum_{i=\kappa-1}^{\kappa'-1} \mu_{\kappa',i}^2 r_{i,i}$, do κ:=$\kappa - 1$.
5. For $i = 1$ to $\kappa - 1$, $\mu_{\kappa,i}$:=$\mu_{\kappa',i}$.
6. Insert $\boldsymbol{b}_{\kappa'}$ right before $\boldsymbol{b}_\kappa$.
7. κ:=$\kappa + 1$.
8. Output $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$.

Fig. 2. The L^3 Algorithm

one usually selects $\delta \approx 1$ and $\eta \approx 1/2$, so that $\|\boldsymbol{b}_1\| \leq (4/3)^{(d-1)/4} \mathrm{vol}(L)^{1/d}$ approximately. The L^3 algorithm obtains in polynomial time a basis reduced with factor $(\delta, 1/2)$ where $\delta < 1$ can be chosen arbitrarily close to 1. The new L^2 algorithm achieves a factor (δ, η), where $\delta < 1$ can be arbitrarily close to 1 and $\eta > 1/2$ arbitrarily close to $1/2$. It is unknown whether or not $\delta = 1$ can be achieved in polynomial time, and whether or not $\eta = 1/2$ can be achieved in quadratic time like L^2. The case $(\delta, \eta) = (1, 1/2)$ is closely related to a notion of reduction invented by Hermite [13] in the language of quadratic forms.

The L^3 algorithm. The L^3 algorithm [19] is described in Fig. 2. It computes an L^3-reduced basis in an iterative fashion: the index κ is such that at any stage of the algorithm, the truncated basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\kappa-1}]$ is L^3-reduced. At each loop iteration, κ is either incremented or decremented: the loop stops when κ reaches the value $d+1$, in which case the entire basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ is L^3-reduced. L^3 performs two kinds of operations: swaps of consecutive vectors and Babai's nearest plane algorithm [3] (see Fig. 3), which uses at most d translations of the form $\boldsymbol{b}_\kappa$:=$\boldsymbol{b}_\kappa - m\boldsymbol{b}_i$, where m is some integer and $i < \kappa$. Swaps are used to achieve Lovász's conditions, while Babai's algorithm is used to size-reduce vectors. We explain Steps 4–7: if Lovász's condition is satisfied, nothing happens in Steps 5 and 6, and κ is incremented like in classical descriptions of the L^3 algorithm. Otherwise, Step 4 finds the right index to insert $\boldsymbol{b}_\kappa$, thus collecting successive failures of Lovász's test.

If L^3 terminates, it is clear that the output basis is L^3-reduced. What is less clear *a priori* is why L^3 has a polynomial-time complexity. A standard argument

Input: A basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$, its GSO and an index κ.
Output: The basis with $\boldsymbol{b}_\kappa$ size-reduced and the updated GSO.
1. For $i = \kappa - 1$ downto 1 do
2. $\quad \boldsymbol{b}_\kappa$:=$\boldsymbol{b}_\kappa - \lceil \mu_{\kappa,i} \rfloor \boldsymbol{b}_i$.
3. $\quad$ For $j = 1$ to i do
4. $\quad\quad \mu_{\kappa,j}$:=$\mu_{\kappa,j} - \lceil \mu_{\kappa,i} \rfloor \mu_{i,j}$.

Fig. 3. Babai's algorithm to size-reduce $\boldsymbol{b}_\kappa$, so that $|\mu_{\kappa,i}| \leq 1/2$ for all $i < \kappa$

shows that each swap decreases the quantity $\Delta = \prod_{i=1}^{d} \|\boldsymbol{b}_i^*\|^{2(d-i+1)}$ by at least a factor $\delta < 1$, while $\Delta \geq 1$ because the $\boldsymbol{b}_i$'s are integer vectors and Δ can be viewed as a product of squared volumes of lattices spanned by some of the $\boldsymbol{b}_i$'s. This proves that there can be no more than $O(d^2 \log B)$ swaps, and therefore loop iterations, where B is an upper bound on the norms of the input basis vectors. It remains to estimate the cost of each loop iteration. This cost turns out to be dominated by $O(dn)$ arithmetic operations on the basis matrix and GSO coefficients $\mu_{i,j}$ and $r_{i,i}$ which are rational numbers of bit-length $O(d \log B)$. Thus, the overall complexity of the L^3 algorithm described in Fig. 2 without fast integer arithmetic is $O((d^2 \log B) \cdot dn \cdot (d \log B)^2)) = O(d^5 n \log^3 B)$.

L^3 with *fpa*. The cost of L^3 is dominated by arithmetic operations on the GSO coefficients which are rationals with huge numerators and denominators. It is therefore tempting to replace the exact GSO coefficients by *fp* approximations. But doing so in a straightforward manner leads to instability. The algorithm is no longer guaranteed to be polynomial-time: it may not even terminate, because the quantity Δ used to bound the complexity of L^3 no longer necessarily decreases at each swap. And if ever the algorithm terminates, the output basis may not be L^3-reduced, due to potential inaccuracy in the GSO coefficients. Prior to this work, the only provable *fp*-L^3 was the one of Schnorr [34], which simulates the behavior of L^3 using *fp*-approximations of the coefficients of the inverse matrix of the $\mu_{i,j}$'s: it computes a $(0.95, 0.55)$-L^3-reduced basis. The number of loop iterations and the number of arithmetic operations (in each iteration) remain the same as L^3: only the cost of each arithmetic operation related to the GSO decreases. Instead of handling integers of length $O(d \log B)$, [34] uses *fp*-numbers with $O(d + \log B)$-bit long mantissæ (with large hidden constants, as mentioned in the introduction), which decreases the worst-case complexity of L^3 to $O(d^4 \log B(d + \log B)^2)$. This is still cubic in $\log B$. Because this algorithm is mostly of theoretical interest, the main number theory computer packages [41, 22, 4] only implement heuristic *fp*-variants of L^3 à la Schnorr-Euchner [36] which suffer from stability problems in high dimension.

3 The L^2 Algorithm

We now describe the L^2 algorithm, which is a natural *fp*-variant of L^3. The main principle is to keep sufficiently good *fp*-approximations of the GSO coefficients during the execution of the algorithm. Accuracy is crucial for size-reduction and for checking Lovász's conditions. If one is not careful, swaps and translations may decrease the accuracy to the point of having meaningless *fp*-values.

3.1 Gram-Schmidt Computations

It is important for L^2 to have accurate formulas for the computation of the GSO coefficients. In [36], the following recursive formulas are used:

$$\mu_{i,j} = \frac{\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle - \sum_{k=1}^{j-1} \mu_{j,k} \cdot \mu_{i,k} \cdot \|\boldsymbol{b}_k^*\|^2}{\|\boldsymbol{b}_j^*\|^2} \text{ and } \|\boldsymbol{b}_i^*\|^2 = \|\boldsymbol{b}_i\|^2 - \sum_{j=1}^{i-1} \mu_{i,j}^2 \cdot \|\boldsymbol{b}_j^*\|^2.$$

In these formulas, the inner products $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ are computed in *fpa*, which leads to a potential inaccuracy of $2^{-\ell}\|\boldsymbol{b}_i\|\|\boldsymbol{b}_j\|$, with the following drawback: to ensure that the basis returned by L^2 is size-reduced, absolute error bounds on the $\mu_{i,j}$'s are required; if the error is therefore larger than $2^{-\ell}\|\boldsymbol{b}_i\|\|\boldsymbol{b}_j\|$, the precision ℓ must be $\Omega(\log B)$ in the worst case. The analyses of [34, 35] do not tackle this issue. We use slightly different formulas by introducing the quantity $r_{i,j} = \mu_{i,j}\|\boldsymbol{b}_j^*\|^2 = \langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle$ for all $i \geq j$:

$$r_{i,j} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle - \sum_{k=1}^{j-1} \mu_{j,k} \cdot r_{i,k} \quad \text{and} \quad \mu_{i,j} = \frac{r_{i,j}}{r_{j,j}}.$$

Accuracy is improved because the inner products are extracted from the exact Gram matrix and because each term of the sum now only requires one multiplication instead of two. For $i = j$, the first formula is $r_{i,i} = \|\boldsymbol{b}_i\|^2 - \sum_{k=1}^{i-1} \mu_{i,k} \cdot r_{i,k}$, which suggests to define $s_j = \|\boldsymbol{b}_i\|^2 - \sum_{k=1}^{j-1} \mu_{i,k} \cdot r_{i,k}$ for all $1 \leq j \leq i$, so that $\|\boldsymbol{b}_i^*\|^2 = r_{i,i} = s_i$. The quantities s_i will be useful to check consecutive Lovász's conditions. Indeed, Lovász's condition $(\delta - \mu_{\kappa,\kappa-1}^2)\|\boldsymbol{b}_{\kappa-1}^*\|^2 \leq \|\boldsymbol{b}_\kappa^*\|^2$ can be rewritten as $\delta\|\boldsymbol{b}_{\kappa-1}^*\|^2 \leq \|\boldsymbol{b}_\kappa^*\|^2 + \mu_{\kappa,\kappa-1}^2\|\boldsymbol{b}_{\kappa-1}^*\|^2$,i.e.,

$$\delta r_{\kappa-1,\kappa-1} \leq s_{\kappa-1}.$$

Whenever the condition is not satisfied, L^3 would swap $\boldsymbol{b}_{\kappa-1}$ and $\boldsymbol{b}_\kappa$, and check the following Lovász's condition:

$$\delta r_{\kappa-2,\kappa-2} \leq s_{\kappa-2}.$$

Thus, storing the s_j's enables us to check consecutive Lovász's conditions (when consecutive swaps occur) without any additional cost since they appear in the calculation of $r_{\kappa,\kappa}$. The computation of $r_{i,j}$, $\mu_{i,j}$ and s_j is summarized in the so-called Cholesky Factorisation Algorithm (CFA) of Fig. 4. Of course, because one

Input: The Gram matrix of $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$.
Output: All the $r_{i,j}$'s, $\mu_{i,j}$'s and s_j's.
1. For $i = 1$ to d do
2. For $j = 1$ to i do
3. $r_{i,j} := \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$,
4. For $k = 1$ to $j - 1$ do $r_{i,j} := r_{i,j} - r_{i,k}\mu_{j,k}$,
5. $\mu_{i,j} := r_{i,j}/r_{j,j}$.
6. $s_0 := \|\boldsymbol{b}_d\|^2$. For $j = 1$ to d do $s_j := s_{j-1} - \mu_{d,j}r_{d,j}$.
7. $r_{d,d} := s_d$.

Fig. 4. The Cholesky Factorisation Algorithm (CFA)

uses *fpa*, the exact values are unknown. Instead, one computes *fp*-approximations $\bar{r}_{i,j}$, $\bar{\mu}_{i,j}$ and $\bar{s}_i$. Steps 4–6 are performed in the following way:

$$\bar{r}_{i,j} := \diamond\left(\bar{r}_{i,j} - \diamond(\bar{r}_{i,k} \cdot \bar{\mu}_{j,k})\right), \ \bar{\mu}_{i,j} := \diamond\left(\bar{r}_{i,j}/\bar{r}_{j,j}\right) \text{ and } \bar{s}_j := \diamond\left(\bar{s}_{j-1} - \diamond(\bar{\mu}_{d,j} \cdot \bar{r}_{d,j})\right).$$

Input: A factor $\eta > 1/2$, a *fp*-precision ℓ, an integer κ, a basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$, $G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$, and *fp* numbers $\bar{r}_{i,j}$ and $\bar{\mu}_{i,j}$'s for $j \leq i < \kappa$.
Output: *fp* numbers $\bar{r}_{\kappa,j}$, $\bar{\mu}_{\kappa,j}$ and $\bar{s}_j$ for $j \leq \kappa$, $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\kappa-1}, \boldsymbol{b}'_\kappa, \boldsymbol{b}_{\kappa+1}, \ldots, \boldsymbol{b}_d]$, and $G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\kappa-1}, \boldsymbol{b}'_\kappa, \boldsymbol{b}_{\kappa+1}, \ldots, \boldsymbol{b}_d)$ where $\boldsymbol{b}'_\kappa = \boldsymbol{b}_\kappa - \sum_{i<\kappa} x_i \boldsymbol{b}_i$ for some integers x_i's and: $|\langle \boldsymbol{b}'_\kappa, \boldsymbol{b}^*_i \rangle| \leq \eta \|\boldsymbol{b}^*_i\|^2$ for any $i < \kappa$.

1. $\bar{\eta} := \frac{\eta+1/2}{2}$. Repeat
2. Compute the $\bar{r}_{\kappa,j}$'s, $\bar{\mu}_{\kappa,j}$'s, $\bar{s}_j$'s with Steps 2–7 of the CFA with "$i = \kappa$".
3. For $i = \kappa - 1$ downto 1 do
4. If $|\bar{\mu}_{\kappa,i}| \geq \bar{\eta}$, then $X_i := \lfloor \bar{\mu}_{\kappa,i} \rceil$, else $X_i := 0$,
5. For $j = 1$ to $i - 1$, $\bar{\mu}_{\kappa,j} := \diamond(\bar{\mu}_{\kappa,j} - \diamond(X_i \cdot \bar{\mu}_{i,j}))$.
6. Update $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ and $G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$, according to $\boldsymbol{b}_\kappa := \boldsymbol{b}_\kappa - \sum_{i=1}^{\kappa-1} X_i \boldsymbol{b}_i$.
7. Until all X_i's are zero.

Fig. 5. The Iterative Babai Nearest Plane Algorithm

We will not use CFA directly in L^2. Instead, we will use parts of it during the execution of the algorithm: because the orthogonalisation is performed vector by vector, there is no need recomputing everything from scratch if the $r_{i,j}$'s and $\mu_{i,j}$'s are already known for i and j below some threshold. Notice that the $r_{i,j}$'s can be updated at the same location, except for $r_{d,d}$ because the different s_j's need being returned to check Lovász's conditions. The CFA will prove useful to estimate in Section 4 the precision required to guarantee the correctness of L^2.

3.2 An Iterative Floating-Point Version of Babai's Algorithm

The core of L^2 is an iterative *fp*-version of Babai's nearest plane algorithm, described in Fig. 5. Instead of size-reducing $\boldsymbol{b}_\kappa$ at once like in Fig. 3, our *fp*-version of Babai performs an iterative process using parts of the CFA algorithm of Fig. 4. Here, the x_i's of Babai's algorithm are computed progressively: the most significant bits in the first loop iteration, then more bits in the second iteration, and so on. This has two nice properties: first it terminates and gives correct results, and, more importantly, it makes L^2 efficient because very few bits of the GSO are required. At Step 4, $\bar{\eta} = \frac{\eta+1/2}{2} \in (1/2, \eta)$ is used instead of η to take into account the fact that $\mu_{\kappa,i}$ is known only approximately. At Step 6, it suffices to update the scalar products $\langle \boldsymbol{b}_i, \boldsymbol{b}_\kappa \rangle$ for $i \leq d$ according to the following relations:

$$\|\boldsymbol{b}'_\kappa\|^2 = \|\boldsymbol{b}_\kappa\|^2 + \sum_{j \neq \kappa} x_j^2 \|\boldsymbol{b}_j\|^2 - 2 \sum_{j \neq \kappa} x_j \langle \boldsymbol{b}_j, \boldsymbol{b}_\kappa \rangle + 2 \sum_{j \neq \kappa, i \neq \kappa} x_i x_j \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$$

$$\langle \boldsymbol{b}_i, \boldsymbol{b}'_\kappa \rangle = \langle \boldsymbol{b}_i, \boldsymbol{b}_\kappa \rangle - \sum_{j \neq \kappa} x_j \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle \quad \text{for} \quad i \neq \kappa.$$

3.3 Main Results

A description of L^2 is given in Fig. 6. There is no need keeping approximations of all the GSO coefficients: because L^2 is iterative, it suffices to have approximations

Input: A valid pair (δ, η) like in Th. 1, a basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ and a *fp*-precision ℓ.
Output: An L^3-reduced basis with factor pair (δ, η).
Variables: A matrix G, two $d \times d$ *fp*-matrices $(\bar{r}_{i,j})$ and $(\bar{\mu}_{i,j})$, a *fp*-vector $\bar{s}$.
1. Compute exactly $G = G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$.
2. $\bar{\delta}$:=$\frac{\delta+1}{2}$, $\bar{r}_{1,1}$:= $\diamond(\langle \boldsymbol{b}_1, \boldsymbol{b}_1 \rangle)$, κ:=2. While $\kappa \leq d$, do
3. Size-reduce $\boldsymbol{b}_\kappa$ using the algorithm of Fig. 5. It updates the *fp*-GSO.
4. κ':=κ. While $\kappa \geq 2$ and $\bar{\delta}\bar{r}_{\kappa-1,\kappa-1} \geq \bar{s}_{\kappa-1}$, do κ:=$\kappa - 1$.
5. For $i = 1$ to $\kappa - 1$ do $\bar{\mu}_{\kappa,i}$:=$\bar{\mu}_{\kappa',i}$, $\bar{r}_{\kappa,i}$:=$\bar{r}_{\kappa',i}$, $\bar{r}_{\kappa,\kappa}$:=$\bar{s}_\kappa$.
6. Insert $\boldsymbol{b}_{\kappa'}$ right before $\boldsymbol{b}_\kappa$ and update G accordingly.
7. κ:=$\kappa + 1$.
8. Output $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$.

Fig. 6. The L^2 algorithm

up to the threshold κ. Notice that the cost of the first step is bounded by $O(d^2 n \log^2 B)$ and is thus negligible compared to the rest of the reduction. At Step 4, $\bar{\delta} = \frac{\delta+1}{2} \in]\delta, 1[$ is used instead of δ to take into account the fact that $\bar{r}_{\kappa-1,\kappa-1}$ and $\bar{s}_{\kappa-1}$ are known only approximately. The main result of this paper is the following:

Theorem 1. *Let (δ, η) such that $1/4 < \delta < 1$ and $1/2 < \eta < \sqrt{\delta}$. Let $c > \log \frac{(1+\eta)^2}{\delta - \eta^2}$ be a constant. Given as input a d-dimensional lattice basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ in $\mathbb{Z}^n$ with $max_i \|\boldsymbol{b}_i\| \leq B$, the L^2 algorithm of Fig. 6 with precision $\ell = cd + o(d)$ outputs a (δ, η)-L^3-reduced basis in time $O(d^4 n \log B(d + \log B))$. More precisely, if τ denotes the number of loop iterations, then the running time is $O(d^2 n(\tau + d \log dB)(d + \log B))$.*

Let us make a few remarks. First, L^2 decreases the complexity bound $O(d^3 n \log B(d + \log B)^2)$ of [34] by a factor $\frac{d + \log B}{d}$. We can choose δ arbitrarily close to 1 and η arbitrarily close to $1/2$, so that the coefficient $c > \log \frac{(1+\eta)^2}{\delta - \eta^2}$ becomes arbitrarily close to $\log 3 < 1.585$. The additional statement related to the number of loop iterations is useful for certain lattices which arise in practice, like lattices arising in knapsacks and minimal polynomials, where $\tau = O(d \log B)$ instead of the usual $O(d^2 \log B)$. Finally, the $o(d)$ term in the condition $\ell = c \cdot d + o(d)$ can be made effective and may be used backwards: if we perform calculations with a fixed number of bits, the correctness will be guaranteed up to a computable dimension.

4 Correctness of the L^2 Algorithm

To guarantee the correctness of L^2, we need to estimate the accuracy of the *fp*-approximations at various stages of the algorithm.

4.1 Accuracy of Gram-Schmidt Computations

In general, the classical Gram-Schmidt algorithm is known to have very poor numerical stability [5, 11, 18, 43]. However, it must be stressed that in the context

of L^3, bases are reduced in an iterative fashion, which implies that we can study the accuracy of Gram-Schmidt computations under the hypothesis that the first $d-1$ vectors of the input basis are L^3-reduced. In this particular setting, because an L^3-reduced basis is roughly orthogonal, the following result shows that a working precision of $\approx d \log 3$ bits is sufficient for the CFA if the reduction factor (δ, η) is sufficiently close to the optimal pair $(1, 1/2)$.

Theorem 2. *Let (δ, η) be a valid factor pair like in Th. 1. Let $\rho = \frac{(1+\eta)^2}{\delta - \eta^2}$. Let $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ in $\mathbb{Z}^n$ be a d-dimensional lattice basis whose Gram matrix is given as input to the CFA algorithm from Fig. 4. Suppose $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d-1}]$ is (δ, η)-L^3-reduced. In the case of* fpa *with a precision ℓ satisfying $d\rho^d 2^{-\ell+2} \leq 1$, the* fp*-numbers returned by the CFA of Fig. 4 satisfy the following equations: for all $j \leq i < d$,*

$$|\bar{r}_{i,j} - r_{i,j}| \leq d\rho^{j-1} 2^{-\ell+4} \cdot r_{j,j} \quad \textit{and} \quad |\bar{\mu}_{i,j} - \mu_{i,j}| \leq d\rho^{j-1} 2^{-\ell+6}.$$

Moreover, if $M = max_{j<d}|\mu_{d,j}|$, then we have for any $j < d$:

$$|\bar{r}_{d,j} - r_{d,j}| \leq d\rho^{j-1} M 2^{-\ell+4} \cdot r_{j,j} \quad \textit{and} \quad |\bar{\mu}_{d,j} - \mu_{d,j}| \leq d\rho^{j-1} M 2^{-\ell+6}.$$

Finally, if $\boldsymbol{b}_d$ is η-size-reduced with respect to $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d-1}]$, then for any $j \leq d$:

$$|\bar{s}_j - s_j| \leq d\rho^{j-1} 2^{-\ell+7} \cdot r_{j,j} + d2^{-\ell} \cdot s_j.$$

The second set of inequalities is useful for the analysis of Babai's algorithm, while the last set provides guarantees when checking Lovász's conditions. We now give a sketch of the proof of Theorem 2 for the case $\eta \approx 1/2$ and $\delta \approx 1$. Most of the accuracy loss comes from Step 4, which amplifies the error. We define $err_j = \max_{i<d} \frac{|\bar{r}_{i,j} - r_{i,j}|}{r_{j,j}}$, i.e., the error on $r_{i,j}$ relative to $r_{j,j}$, and we estimate its growth as j increases. Obviously $err_1 \leq 2^{-\ell} \max_{i<d} \frac{|\langle \boldsymbol{b}_i, \boldsymbol{b}_1 \rangle|}{\|\boldsymbol{b}_1\|^2} \leq 2^{-\ell}$, because of size-reduction. We now choose $j \in [2, d-1]$. The result for $i = d$ can be derived from the proof for $i \leq d-1$, intuitively by replacing "$\boldsymbol{b}_d$" by "$\frac{1}{M}\boldsymbol{b}_d$" in it. Because of Step 5, for any $i < d$ and any $k < j$:

$$|\bar{\mu}_{i,k} - \mu_{i,k}| \leq \left| \frac{r_{k,k}}{\bar{r}_{k,k}} \right| err_k + |r_{i,k}| \left| \frac{1}{\bar{r}_{k,k}} - \frac{1}{r_{k,k}} \right| \leq \left(\frac{3}{2} + \epsilon \right) err_k,$$

where we neglected low-order terms and used the fact that $|r_{i,k}| \leq \left(\frac{1}{2} + \epsilon\right) \|\boldsymbol{b}_k\|^2$, which comes from size-reduction. This implies that:

$$\begin{aligned} |\diamond(\bar{\mu}_{j,k} \cdot \bar{r}_{i,k}) - \mu_{j,k} r_{i,k}| &\leq |\bar{\mu}_{j,k} - \mu_{j,k}| \cdot |\bar{r}_{i,k}| + |\mu_{j,k}| \cdot |\bar{r}_{i,k} - r_{i,k}| \\ &\leq \left(\frac{5}{4} + \epsilon \right) err_k \cdot \|\boldsymbol{b}_k^*\|^2, \end{aligned}$$

where we also neglected low-order terms and used size-reduction twice. Thus,

$$err_j \leq \left(\frac{5}{4} + \epsilon \right) \sum_{k<j} \frac{\|\boldsymbol{b}_k^*\|^2}{\|\boldsymbol{b}_j^*\|^2} err_k \leq \left(\frac{5}{4} + \epsilon \right) \sum_{k<j} \left(\frac{4}{3} + \epsilon \right)^{j-k} err_k,$$

by using Lovász's conditions. This last inequality finally gives $err_j \leq (3+\epsilon)^j \cdot err_1 \leq (3+\epsilon)^j 2^{-\ell}$, since we have $\left(\frac{4}{3}+\epsilon\right)\left(\frac{5}{4}+\epsilon+1\right) \approx 3+\epsilon$. □

The bound in Theorem 2 seems to be tight, at least in practice: the classical Gram-Schmidt algorithm or the CFA become experimentally inaccurate with a precision $\leq d \log 3$ bits for certain bases. Consider indeed the L^3-reduced lattice basis given by the rows of the following $d \times d$ matrix L:

$$\begin{array}{lll} L_{i,i} = & (\sqrt{4/3})^{d-i} & \\ L_{i,j} = (-1)^{i-j+1} L_{i,i} \cdot \text{random}[0.49, 0.5] & \text{if} & j > i \\ L_{i,j} = \quad 0 & \text{if} & j < i. \end{array}$$

To obtain an integral lattice, one can multiply L by a large scaling factor and round its entries. By definition, this matrix is already L^3-reduced. With double precision calculations, i.e., with 53-bit mantissæ, the error on the $\mu_{i,j}$'s becomes significant (higher than 0.5) in dimension 35. These bases show the tightness of our $\log 3 \cdot d$ bound. By inserting a suitable random vector to such a basis, we were able to make the LLL_FP routine of NTL loop forever in dimension 55 (see [28]). This invalidates the claim of [36, 37, 15] which states that double precision suffices for lattices of dimension up to ≈ 250 using classical Gram-Schmidt.

4.2 Accuracy of Babai's Nearest Plane Algorithm

To estimate the accuracy of the iterative *fp*-version of Babai's algorithm given in Fig. 5 and used in L^2, we first study a simpler *fp*-version described in Fig. 7.

Input: A *fp*-precision ℓ, a basis $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$, $G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$ and *fp* numbers $\bar{r}_{i,j}$'s and $\bar{\mu}_{i,j}$'s for $j \leq i < d$.
Output: $x_1, \ldots, x_{d-1} \in \mathbb{Z}$ and $G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d-1}, \boldsymbol{b}'_d)$, where $\boldsymbol{b}'_d = \boldsymbol{b}_d - \sum_{i<d} x_i \boldsymbol{b}_i$.
1. Compute the $\bar{\mu}_{d,j}$'s for $j < d$ with Steps 2–7 of the CFA with "$i = d$".
2. $\bar{\eta} := \frac{\eta + 1/2}{2}$. For $i = d-1$ downto 1 do
3. If $|\bar{\mu}_{d,i}| \geq \bar{\eta}$, then $x_i := \lfloor \bar{\mu}_{d,i}^{(i+1)} \rceil$, else $x_i := 0$,
4. For $j = 1$ to $i-1$ do $\bar{\mu}_{d,j} := \diamond\left(\bar{\mu}_{d,j} - \diamond(x_i \cdot \bar{\mu}_{i,j})\right)$.
5. Compute $G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d-1}, \boldsymbol{b}'_d)$ from $G(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d-1}, \boldsymbol{b}_d)$.

Fig. 7. Babai's Nearest Plane Algorithm

We use Theorem 2 to show stability properties of Babai's algorithm:

Theorem 3. *Let (δ, η) be a valid reduction factor (like in Th. 1) and $\rho = \frac{(1+\eta)^2}{\delta - \eta^2}$. Let $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ in $\mathbb{Z}^n$ be a d-dimensional lattice basis given as input to the algorithm of Fig. 7, and $B = max_i \|\boldsymbol{b}_i\|$. Suppose that $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{d-1}]$ is (δ, η)-L^3-reduced and that the given $\bar{r}_{i,j}$'s and $\bar{\mu}_{i,j}$'s are those that would have been returned by the CFA with working precision ℓ. Let $M = max_j |\mu_{d,j}|$. If ℓ satisfies $d^2 \rho^d 2^{-\ell+2} \leq 1$, the algorithm of Fig. 7 finds integers $x_1, \ldots, x_{d-1}$ such that for any $i < d$:*

$$|x_i| \leq 2(1+\eta)^{d-1-i}(M+1) \quad \textit{and} \quad \frac{|\langle \boldsymbol{b}'_d, \boldsymbol{b}^*_i \rangle|}{\|\boldsymbol{b}^*_i\|^2} \leq \bar{\eta} + d\rho^d(M+1)2^{-\ell+4}.$$

Moreover it works in time $O(dn\ell(d+\log B))$ *as long as* $\ell = O(d+\log B)$.

Notice that by using the relation $\log M = O(d+\log B)$ (coming from the fact that the $d-1$ first vectors are L^3-reduced), this result implies that taking $\ell = O(d+\log B)$ is sufficient to make the $|\mu_{d,i}|$'s smaller than η. The drawback of this approach is that one should have previously computed the $r_{i,j}$'s and $\mu_{i,j}$'s with precision $O(d+\log B)$. This seems an overkill since $O(d+\log M)$ bits suffice and M is usually far smaller than B. Indeed, in the case of the Euclid algorithm, the analogy is that the quotients would be computed by using all the bits of the remainders, instead of only the most significant ones.

The iterative Babai algorithm from Fig. 5 is a way to work around the difficulty that M cannot be tightly bounded in advance. Using only a $O(d)$-bit precision, it finds the x_j's progressively by performing successive Babai steps, each one making $\log M$ decrease by $\Omega(d)$, until we reach $M \leq \eta$. This strategy is somewhat similar to the Babai routine of the floating-point L^3 algorithm of NTL [41], which repeatedly applies Babai's algorithm until nothing happens.

The iterative Babai algorithm will use a precision $\ell = \left(\log \frac{(1+\eta)^2}{\delta-\eta^2} + C\right) d + o(d)$ with an arbitrary $C > 0$. The CFA with working precision ℓ gives the input $\bar{r}_{i,j}$'s and $\bar{\mu}_{i,j}$'s, which by Theorem 2 have their $\approx Cd$ leading bits correct. Therefore, the $r_{i,j}$'s and $\mu_{i,j}$'s may not be known sufficiently well to perform Babai's algorithm in one single step, but Theorem 3 gives that their approximations suffice to make $M = \max_{i<\kappa} \frac{|\langle \boldsymbol{b}_\kappa, \boldsymbol{b}^*_i \rangle|}{\|\boldsymbol{b}^*_i\|^2}$ decrease by $\approx Cd$ bits. By making $O\left(1 + \frac{\log M}{d}\right)$ such calls to Babai's algorithm, size-reduction can be achieved.

Theorem 4. *Let* (δ, η) *be a valid pair (like in Th. 1) and* $\rho = \frac{(1+\eta)^2}{\delta-\eta^2}$. *Let* $C > 0$ *be a constant. Let* $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d]$ *be a* d*-dimensional lattice basis in* $\mathbb{Z}^n$ *given as input to the algorithm of Fig. 7, and* $B = max_i \|\boldsymbol{b}_i\|$. *Suppose that* $[\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\kappa-1}]$ *is* (δ, η)*-*L^3*-reduced and that the given* $\bar{r}_{i,j}$*'s,* $\bar{\mu}_{i,j}$*'s are those that would have been returned by the CFA with precision* ℓ. *Let* $M = max_{j<\kappa}|\mu_{\kappa,j}|$. *If* ℓ *satisfies* $d^2\rho^d 2^{-\ell+6+Cd} \leq \eta - \frac{1}{2}$, *the algorithm of Fig. 5 provides a correct output and the returned* $\bar{r}_{\kappa,j}$*'s,* $\bar{\mu}_{\kappa,j}$*'s,* $\bar{s}_j$*'s are those that would have been returned by the CFA with precision* ℓ. *Moreover, if* $\ell = O(d)$, *then the running time is* $O(dn(d+\log B)(d+\log M))$.

Proof. We start by the correctness properties of the algorithm. At the last iteration of the main loop, the computed X_j's are all zero, which implies that nothing happens during Steps 3–6. This gives that for any $j < \kappa$, $|\bar{\mu}_{\kappa,j}| \leq \eta^-$, from which we derive $\frac{|\langle \boldsymbol{b}'_\kappa, \boldsymbol{b}^*_i \rangle|}{\|\boldsymbol{b}^*_i\|^2} \leq \eta$, by using Theorem 3 and the hypothesis on ℓ. This also gives the correctness of the returned $\bar{r}_{\kappa,j}$'s, $\bar{\mu}_{\kappa,j}$'s and $\bar{s}_j$'s.

We now consider the impact of one iteration of the main loop on M. Let M_1 be the "new M" after the loop iteration. Theorem 3 and the hypothesis on ℓ give the inequality:

$$M_1 \leq \bar{\eta} + d\rho^d(1+M)2^{-\ell+4} \leq \frac{1/2+2\eta}{3} + 2^{-Cd}M.$$

As a consequence, there can be at most $1+\frac{1}{Cd}(\log\frac{\eta-1/2}{3}+\log M)$ loop iterations.

Theorem 3 gives that the cost of one loop iteration is bounded by $O(d^2n(d+\log B))$, because during the execution of the algorithm, the entries of the Gram matrix remain integers of length bounded by $O(d+\log B)$. The fact that we have additional vectors in the basis (namely $\boldsymbol{b}_{\kappa+1},\ldots,\boldsymbol{b}_d$) is taken into account in the complexity bound. Finally, the overall cost of the algorithm is bounded by:

$$O\left(d^2n(d+\log B)\left(1+\tfrac{\log M}{d}\right)\right) = O(dn(d+\log B)(d+\log M)). \qquad \square$$

4.3 Application to L^2

We now prove the correctness of L^2. To do this, we show the following:

Theorem 5. *Let $[\boldsymbol{b}_1^{(0)},\ldots,\boldsymbol{b}_d^{(0)}]$ in $\mathbb{Z}^n$ be a lattice basis given as input to the L^2 algorithm. For any loop iteration t, let $[\boldsymbol{b}_1^{(t)},\ldots,\boldsymbol{b}_d^{(t)}]$ denote the current basis at the beginning of the t-th iteration. We have:*

1. *For any $i \leq \kappa(t)-1$, $\boldsymbol{b}_i^{(t)}$ is η-size-reduced, and $[\boldsymbol{b}_1^{(t)},\ldots,\boldsymbol{b}_{\kappa(t)-1}^{(t)}]$ is L^3-reduced with factor pair (δ,η).*
2. *For any $i \leq d$, $max_{j\leq i}\|\boldsymbol{b}_j^{(t)*}\| \leq max_{j\leq i}\|\boldsymbol{b}_j^{(0)*}\|$ and $\|\boldsymbol{b}_i^{(t)}\| \leq \sqrt{d}max_{j\leq i}\|\boldsymbol{b}_j^{(0)}\|$.*

Proof. Clearly, all these properties are valid for $t=0$, and size-reduction comes from Theorem 4. Assume now that we are performing the t-th loop iteration.

We now show that Lovász's tests work as desired. Recall that $\boldsymbol{b}_\kappa$ is swapped with $\boldsymbol{b}_i$ for $i<\kappa$ if and only if for any $j\in[i,\kappa-1]$ we have $\bar{s}_j \leq \bar{\delta}\bar{r}_{j,j}$. Assume first that $\boldsymbol{b}_\kappa$ is swapped with $\boldsymbol{b}_j$. Theorem 2 gives that:

$$s_j(1-d2^{-\ell}) \leq \bar{r}_{j,j}(\bar{\delta}+d\rho^{j-1}2^{-\ell+8}).$$

Therefore, as soon as $d\rho^d 2^{-\ell+5} \leq 1-\delta$, if a swap is done in L^2 then Lovász's condition was not fulfilled for the factor $\frac{2-\delta}{3} \in (\delta,1)$. On the opposite, assume that $\boldsymbol{b}_\kappa$ and $\boldsymbol{b}_j$ are not swapped. Theorem 2 gives:

$$s_j(1+d2^{-\ell}) \geq r_{j,j}(\bar{\delta}-d\rho^{j-1}2^{-\ell+8}).$$

Thus, as soon as $d\rho^d 2^{-\ell+4} \leq 1-\delta$, if there is no swap, then Lovász's condition was fulfilled for δ. This gives the first statement for the new loop iteration. For the second statement, observe that during a swap between $\boldsymbol{b}_k$ and $\boldsymbol{b}_{k-1}$:

- $\|\boldsymbol{b}_{k-1}^{*new}\| \leq \|\boldsymbol{b}_{k-1}^{*old}\|$ because of Lovász's condition,
- $\|\boldsymbol{b}_k^{*new}\| \leq \|\boldsymbol{b}_{k-1}^{*old}\|$ because $\boldsymbol{b}_k^{*new}$ is an orthogonal projection of $\boldsymbol{b}_{k-1}^{*old}$,

which gives the first part of the second statement. Finally, if $\boldsymbol{b}_i^{(t)}$ appears during the execution of the algorithm and is size-reduced, we have:

$$\|\boldsymbol{b}_i^{(t)}\|^2 \leq d\cdot\max_{j\leq i}\|\boldsymbol{b}_j^{(t)*}\|^2 \leq d\cdot\max_{j\leq i}\|\boldsymbol{b}_j^{(0)*}\|^2 \leq d\cdot\max_{j\leq i}\|\boldsymbol{b}_j^{(0)}\|^2.$$

This proves the second part for $i < \kappa(t)$. If $i \geq \kappa(t)$, we consider the largest $t' < t$ such that $\kappa(t'+1)-1 = i$. The iteration t' was the one which created $\boldsymbol{b}_i^{(t)}$. If t' does not exist, this vector is an initial vector and the result is obvious. Otherwise $\boldsymbol{b}_i^{(t)} = \boldsymbol{b}_{\kappa(t'+1)-1}^{(t'+1)}$ is size-reduced at the $(t'+1)$-th iteration. Thus we have $\|\boldsymbol{b}_i^{(t)}\|^2 \leq d \cdot \max_{j \leq \kappa(t'+1)-1} \|\boldsymbol{b}_j^{(0)}\|^2$. Since κ cannot increase by more than 1 in a single iteration, we must have $i \geq \kappa(t'+1)-1$, which ends the proof.

5 Complexity Analysis of the L² Algorithm

We now prove the complexity statement of Theorem 1.

5.1 On the Number of Loop Iterations of L²

In Section 4.3, we showed that the accuracy suffices to check Lovász's conditions. For any Lovász test, either κ increases or decreases by one and when it decreases, the quantity $\prod_{i=1}^{d} \|\boldsymbol{b}_i^*\|^{2(d-i)}$ decreases by a factor of at least $\frac{3}{2\delta+1} > 1$. It is a standard L^3 argument that this quantity is actually an integer (it is a product of squared volumes of integer lattices) and is initially bounded by $B^{O(d^2)}$. But during the execution of the algorithm, the difference between the numbers of decreases and increases of κ is exactly d, so there are at most $O(d^2 \log B)$ loop iterations.

In the rest of this section we show how to achieve the complexity bound $O(d^4 n \log B(d + \log B))$. This is done by generalizing a cascade phenomenon which appears in the analyses of the Euclidean and Gaussian algorithms.

5.2 Analyses of the Euclidean Algorithm

As mentioned in the introduction, the L^3 algorithm can be viewed as a high-dimensional generalisation of the Euclidean algorithm to compute gcds. But this analogy is incomplete: the Euclidean algorithm has a quadratic complexity bound, whereas the L^3 algorithm is cubic for any fixed dimension. In some sense, the analysis of the standard L^3 algorithm corresponds to a naive analysis of the Euclidean algorithm which also gives a cubic complexity. Recall the Euclidean algorithm: given as input two numbers $r_0 > r_1 > 0$, Euclid successively computes the quotients q_i and remainders r_i defined by $q_i = \lfloor r_{i-1}/r_i \rfloor$ and $r_{i+1} = r_{i-1} - q_i r_i$, until $r_{\tau+1} = 0$ for some τ. Then r_τ is the gcd of r_0 and r_1. It is well-known that the remainders decrease at least geometrically, so that the number of Euclidean divisions is $\tau = O(\log r_0)$. A naive analysis of the Euclidean algorithm states that the algorithm performs $O(\log r_0)$ arithmetic operations on integers of lengths bounded by $O(\log r_0)$, so that the overall cost is bounded by $O(\log^3 r_0)$. A well-known more subtle analysis notices that the cost of computing q_i and r_{i+1} without fast integer arithmetic is bounded by $O(\log r_{i-1} \cdot \log q_i) = O(\log r_0 \cdot (1 + \log r_{i-1} - \log r_i))$. Summed over all the steps, all but two terms "$\log r_i$" vanish, leading to the classical quadratic complexity bound.

This improved Euclidean analysis cannot unfortunately be extended to the standard L^3, because the GSO coefficients are too big. The bit-length of the numerator and denominator of most GSO coefficients is $O(d \log B)$: computing the exact GSO of an L^3-reduced basis already takes cubic time. Surprisingly, the improved Euclidean analysis can be extended to L^2: this would not be possible without a working precision independent of $\log B$. The main difficulty is to generalize the cancellation of all but a very few terms in the sum of the costs of consecutive loop iterations. In the Euclidean and Gaussian algorithms, this cancellation is trivial because two consecutive costs compensate each other directly. The phenomenon is much more complicated in higher dimension: the cost of the t-th iteration will not necessarily be balanced by the cost of the previous $(t-1)$-th iteration, but by the cost of the t'-th iteration for some $t' < t$. Special care must be taken so that the t''s do not collide.

5.3 A Cascade in Arbitrary Dimension

In this subsection we complete the proof of Theorem 1. We already know that the number of loop iterations is $\tau = O(d^2 \log B)$. The t-th loop iteration costs $O(dn(d + \log B)(d + \log M(t)))$ where $M(t) = \max_{j<\kappa(t)} |\mu_{\kappa(t),j}(t)|$. By analogy with the Euclidean algorithm, we make terms cancel out in the sum over the loop iterations of the "$\log M(t)$'s". For this purpose, we define the index $\alpha(t)$ as the smallest swapping index since the last time κ was at least $\kappa(t)$.

Lemma 1. *Let t be a loop iteration. Let $\phi(t) = max(t' < t \mid \kappa(t') \geq \kappa(t))$ if it exists and 1 otherwise, and let $\alpha(t) = min(\kappa(t') \mid t' \in [\phi(t), t)) - 1$. Then we have $\log M(t) \leq d + \log \|\boldsymbol{b}_{\kappa(t)}^{(t)}\| - \log \|\boldsymbol{b}_{\alpha(t)}^{(t)}\|$.*

We are to subdivide the sum of the $\log M(t)$'s over the successive loop iterations into $O(d)$ subsums according to the value of $\kappa(t)$:

$$\sum_{t \leq \tau} \left[d + \log \|\boldsymbol{b}_{\kappa(t)}^{(t)}\| - \log \|\boldsymbol{b}_{\alpha(t)}^{(t)}\| \right] \leq \tau d + \sum_{k=2}^{d} \sum_{\{t \mid \kappa(t)=k\}} \left[\log \|\boldsymbol{b}_{k}^{(t)}\| - \log \|\boldsymbol{b}_{\alpha(t)}^{(t)}\| \right].$$

For each of these subsums, we keep $k-1$ positive terms and $k-1$ negative terms, and make the others vanish in a progressive cancellation. Terms proportional to d can appear from such cancellations, but they will be absorbed in "$O(\tau d)$". The crucial point to do this is the following:

Lemma 2. *Let $k \in [2, d]$ and $t_1 < \ldots < t_k$ be loop iterations of the L^2 algorithm such that for any $j \leq k$, $\kappa(t_j) = k$. Then there exists $j < k$ with:*

$$\|\boldsymbol{b}_{\alpha(t_j)}^{(t_j)}\| \geq d(\delta - \eta^2)^{-d/2} \|\boldsymbol{b}_{k}^{(t_k)}\|.$$

To prove this result, we need the following technical fact:

Lemma 3. *Let T and j be integers such that $\kappa(T) \geq j \geq \kappa(T+1)$. We have:*

$$max_{i \leq j} \|\boldsymbol{b}_i^{(T+1)*}\| \leq max_{i<j} \|\boldsymbol{b}_i^{(T)*}\| \quad and \quad max_{i \leq j} \|\boldsymbol{b}_i^{(T+1)}\| \leq \sqrt{d} \cdot max_{i<j} \|\boldsymbol{b}_i^{(T)}\|.$$

It is now possible to finish the complexity analysis. Let $k \in [2, d]$ and $t_1 < \ldots < t_{\tau_k} = \{t \leq \tau \mid k(t) = k\}$. We extract from the global sum the terms corresponding to these loop iterations. Theorem 5 and the fact we are dealing with an integer lattice ensures that:

$$\sum_{i=1}^{\tau_k} \log \frac{\|\boldsymbol{b}_k^{(t_i)}\|}{\|\boldsymbol{b}_{\alpha(t_i)}^{(t_i)}\|} \leq (k-1)\log(\sqrt{d}B) + \sum_{i=k}^{\tau_k} \log \|\boldsymbol{b}_k^{(t_i)}\| - \sum_{i=1}^{\tau_k - k + 1} \log \|\boldsymbol{b}_{\alpha(t_i)}^{(t_i)}\|.$$

Lemma 2 helps to tighly bound the right-hand term above. First, we apply it with $t_1, \ldots, t_k$. This shows that there exists $j < k$ such that $\|\boldsymbol{b}_k^{(t_k)}\| \leq d(\delta - \eta^2)^{-d/2}\|\boldsymbol{b}_{\alpha(t_j)}^{(t_j)}\|$. The indices "$i = k$" in the positive sum and "$i = j$" in the negative sum cancel out and a term "$\frac{d}{2}\log(\delta - \eta^2)^{-1} + \log d$" appears. Then we use Lemma 2 with t_{k+1} and the $k-1$ first t_i's that remain in the negative sum. It is easy to see that t_{k+1} is larger than any of them, so that we can have another "positive-negative" pair which cancels out in a "$\frac{d}{2}\log(\delta - \eta^2)^{-1} + \log d$". We perform this operation $\tau_k - k + 1$ times, to obtain:

$$\sum_{i=1}^{\tau_k} \log \frac{\|\boldsymbol{b}_k^{(t_i)}\|}{\|\boldsymbol{b}_{\alpha(t_i)}^{(t_i)}\|} \leq (k-1)\log(\sqrt{d}B) + \tau_k \left[\frac{d}{2}\log(\delta - \eta^2)^{-1} + \log d\right].$$

The fact that $\sum_k \tau_k = \tau$ finally gives $\sum_{t \leq \tau}(d + \log M(t)) = O(\tau d + d^2 \log(dB))$.

Acknowledgments

We thank Richard Brent, Guillaume Hanrot, Claus Peter Schnorr and Paul Zimmermann for numerous discussions and anonymous referees for comments which improved the presentation of the results.

References

1. LIDIA 2.1.3. A C++ library for computational number theory. `http://www.informatik.tu-darmstadt.de/TI/LiDIA/`.
2. IEEE 754. IEEE standard for binary floating-point arithmetic.
3. L. Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.
4. C. Batut, K. Belabas, D. Bernardi, H. Cohen, and M. Olivier. PARI/GP computer package version 2. Université de Bordeaux I. `http://pari.math.u-bordeaux.fr/`
5. Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, 1996.
6. D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.
7. D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $n^{0.292}$. In *Proc. of Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1999.

8. H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1995. Second edition.
9. D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. of Cryptology*, 10(4):233–260, 1997.
10. O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Proc. of Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer-Verlag, 1997.
11. G. Golub and C. van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 1996.
12. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1993.
13. C. Hermite. Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres, deuxième lettre. *J. Reine Angew. Math.*, 40:279–290, 1850. Also available in pp122–135 of the first volume of Hermite's complete works, published by Gauthier-Villars.
14. N. A. Howgrave-Graham and N. P. Smart. Lattice attacks on digital signature schemes. *Design, Codes and Cryptography*, 23:283–290, 2001.
15. H. Koy and C. P. Schnorr. Segment LLL-reduction of lattice bases. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*, pages 67–80. Springer-Verlag, 2001.
16. H. Koy and C. P. Schnorr. Segment LLL-reduction with floating point orthogonalization. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*, pages 81–96. Springer-Verlag, 2001.
17. J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *Journal of the Association for Computing Machinery*, January 1985.
18. C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. SIAM, 1995.
19. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:513–534, 1982.
20. H. W. Lenstra, Jr. Integer programming with a fixed number of variables. Technical report, Mathematisch Instituut, Universiteit van Amsterdam, April 1981. Report 81-03.
21. H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.
22. Magma. The Magma computational algebra system for algebra, number theory and geometry. `http://www.maths.usyd.edu.au:8000/u/magma/`.
23. D. Micciancio. Improving lattice-based cryptosystems using the Hermite normal form. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
24. D. Micciancio and S. Goldwasser. *Complexity of lattice problems: A cryptographic perspective*. Kluwer Academic Publishers, Boston, 2002.
25. P. Q. Nguyễn. Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97. In *Proc. of the 19th Cryptology Conference (Crypto '99)*, volume 1666 of *Lecture Notes in Computer Science*, pages 288–304. IACR, Springer-Verlag, 1999.
26. P. Q. Nguyễn and I. E. Shparlinski. The insecurity of the Digital Signature Algorithm with partially known nonces. *Journal of Cryptology*, 15(3):151–176, 2002.
27. P. Q. Nguyễn and D. Stehlé. Low-dimensional lattice basis reduction revisited (extended abstract). In *Proc. of ANTS-VI*, volume 3076 of *Lecture Notes in Computer Science*, pages 338–357. Springer-Verlag, 2004. Journal version in preparation.

28. P. Q. Nguyễn and D. Stehlé. A 55-dimensional lattice which makes NTL [41]'s LLL_FP (with $\delta = 0.99$) loop forever. Available at `http://www.loria.fr/~stehle/FPLLL.html`
29. P. Q. Nguyễn and J. Stern. Cryptanalysis of the Ajtai-Dwork Cryptosystem. In *Proc. of the 18th Cryptology Conference (Crypto '98)*, volume 1462 of *Lecture Notes in Computer Science*, pages 223–242. IACR, Springer-Verlag, 1998.
30. P. Q. Nguyễn and J. Stern. The two faces of lattices in cryptology. In *Cryptography and Lattices – Proc. CALC '01*, volume 2146 of *Lecture Notes in Computer Science*, pages 146–180. Springer-Verlag, 2001.
31. A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Cryptology and Computational Number Theory*, volume 42 of *Proc. of Symposia in Applied Mathematics*, pages 75–88. A.M.S., 1990.
32. The SPACES Project. MPFR, a LGPL-library for multiple-precision floating-point computations with exact rounding. `http://www.mpfr.org/`.
33. C. P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Th. Computer Science*, 53:201–224, 1987.
34. C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *J. of algorithms*, 9(1):47–62, 1988.
35. C. P. Schnorr. Fast LLL-type lattice reduction. Unpublished draft available at `http://www.mi.informatik.uni-frankfurt.de/research/papers.html`, October 2004.
36. C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. In *Proc. of FCT '91*, volume 591 of *Lecture Notes in Computer Science*, pages 68–85. Springer-Verlag, 1991.
37. C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math. Programming*, 66:181–199, 1994.
38. A. Schönhage. Factorization of univariate integer polynomials by diophantine aproximation and an improved basis reduction algorithm. In *Proc. of ICALP '84*, Lecture Notes in Computer Science, pages 436–447. Springer-Verlag, 1984.
39. A. Schönhage. Fast reduction and composition of binary quadratic forms. In *Proc. of ISSAC '91*, pages 128–133. ACM Press, 1991.
40. A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.
41. V. Shoup. NTL, Number Theory C++ Library. `http://www.shoup.net/ntl/`.
42. A. Storjohann. Faster algorithms for integer lattice basis reduction. Technical report, ETH Zurich, 1996.
43. J. H. Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, New-York, 1988.
44. C. K. Yap. Fast unimodular reduction: Planar integer lattices. In *Proc. of the 33rd Annual Symposium on Foundations of Computer Science*, IEEE, pages 437–446, 1992.

Practical Cryptography in High Dimensional Tori

Marten van Dijk[1], Robert Granger[2], Dan Page[2], Karl Rubin[3,⋆], Alice Silverberg[3,⋆⋆], Martijn Stam[2], and David Woodruff[1,⋆⋆⋆]

[1] MIT CSAIL
{marten, dpwood}@mit.edu
[2] Department of Computer Science, University of Bristol
{granger, page, stam}@cs.bris.ac.uk
[3] Department of Mathematics, UC Irvine
{krubin, asilverb}@uci.edu

Abstract. At Crypto 2004, van Dijk and Woodruff introduced a new way of using the algebraic tori T_n in cryptography, and obtained an asymptotically optimal $n/\phi(n)$ savings in bandwidth and storage for a number of cryptographic applications. However, the computational requirements of compression and decompression in their scheme were impractical, and it was left open to reduce them to a practical level. We give a new method that compresses orders of magnitude faster than the original, while also speeding up the decompression and improving on the compression factor (by a constant term). Further, we give the first efficient implementation that uses T_{30}, compare its performance to XTR, CEILIDH, and ECC, and present new applications. Our methods achieve better compression than XTR and CEILIDH for the compression of as few as two group elements. This allows us to apply our results to ElGamal encryption with a small message domain to obtain ciphertexts that are 10% smaller than in previous schemes.

Keywords: torus-based cryptography, discrete-log based cryptography.

1 Introduction

When Diffie and Hellman introduced their key agreement scheme in a finite field of prime order, they made the assumption that a birthday attack was the best one can do (the Pohlig-Hellman algorithm [20] was in submission). Hence it made sense to use a subgroup of size about that of the field itself. Since then, the discrete logarithm problem in the multiplicative group of a finite field has been studied with increased interest. For prime order subgroups the best known

⋆ Rubin was supported by NSF grant DMS-0140378.
⋆⋆ Silverberg was supported by NSA grant MDA904-03-1-0033.
⋆⋆⋆ Woodruff was supported by an NDSEG fellowship.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 234–250, 2005.

attacks today are the birthday attack in the subgroup itself and the Number Field Sieve in the full finite field. It is now common practice to use a subgroup whose cardinality is substantially smaller than the field size. This raises the question whether it is possible to efficiently represent elements in this subgroup with fewer bits than generic elements of the full field, thus providing compression.

Brouwer, Pellikaan and Verheul [4] showed that compression can be achieved by going up to an extension field. They conjectured that one can attain a compression ratio of $n/\phi(n)$, where n is the degree of the extension. For $n = 2$, the LUC cryptosystem [14] already achieved these savings. For $n = 6$, Brouwer et al. described a system that was later improved upon by Lenstra and Verheul [15, 16], resulting in the XTR public key cryptosystem.

Rubin and Silverberg [23] recast the problem of compression for extension fields in terms of algebraic tori. They showed that if the algebraic torus T_n is rational, the conjectured compression factor $n/\phi(n)$ can in fact be achieved. If n is the product of at most two prime powers then T_n is known to be rational [30, 12]. Based on the rationality of T_6, Rubin and Silverberg [23] developed the CEILIDH public key cryptosystem.

Although T_n is not known to be rational in general, van Dijk and Woodruff [6] show that one can obtain key agreement, signature and encryption schemes with a compression factor asymptotically $n/\phi(n)$ as the number of keys, signatures, or messages grows, *without* relying on the rationality of T_n. This helps explain the potential of, and increasing interest in, torus-based cryptography.

The torus T_n is rational if there are efficiently computable "almost bijections"[1] between $T_n(\mathbb{F}_q)$ and $\mathbb{F}_q^{\phi(n)}$, where ϕ is Euler's totient function. Though the tori T_n in general are only conjectured to be rational, it is known [30] that they are always *stably rational*, i.e., for every n there is an m such that there is an "almost bijection" between $T_n(\mathbb{F}_q) \times \mathbb{F}_q^m$ and $\mathbb{F}_q^{\phi(n)+m}$.

Using the fact that T_n is stably rational, van Dijk and Woodruff [6] developed bijections between $T_n(\mathbb{F}_q) \times \mathbb{F}_q^m$ and $\mathbb{F}_q^{\phi(n)+m}$ with $m = \sum_{d|n,\ \mu(n/d)=-1} d$, where μ is the Möbius function, leading to asymptotically optimal $n/\phi(n)$ savings in bandwidth and storage. However, a major drawback of their solution is its large computational requirements.

The present work gives a new and efficient construction of bijections between $T_n(\mathbb{F}_q) \times \mathbb{F}_q^m$ and $\mathbb{F}_q^{\phi(n)+m}$ with significantly smaller m than in [6], as well as an optimised implementation when $n = 30$. The latter uses some known techniques [9] to efficiently implement CEILIDH.

Note that $n = 30 = 2 \cdot 3 \cdot 5$ is the next cryptographically interesting case, since its compression is up to 20% better than that of systems based on $n = 6$. In addition to our computational savings, in this case we are able to reduce the original affine surplus $m = 32$ [6] to $m = 2$. As we show, this reduction has immediate practical implications.

[1] The maps may be undefined on a small number of points.

Since we are interested in the practicality of our construction, we perform timings for exponentiations, compression and decompression for both the new $T_{30}(\mathbb{F}_q)$ system and for a CEILIDH-based $T_6(\mathbb{F}_{q_L})$ system with $q_L \approx q^5$. For an equivalent of 1024-bit RSA security, the computational costs of the operations in both systems are comparable, while the compression of our scheme is better by a factor of $5/4 = (30/\phi(30))/(6/\phi(6))$.

Observe that the security of ECC systems is based on a different mathematical problem and (still) has the advantage that the best known attack is via a birthday paradox argument. So, one works in a subgroup that is almost the same size as the elliptic curve itself. Consequently, compression is much less of an issue there. Yet the same ECC systems provide an additional reason to study the compression methods we do, since the rise of pairing-based cryptography leads to elements in the torus (although at present no efficient curves with embedding degree 30 are known). Note that compression for elliptic curves over extension fields has also been considered [22].

Outline: §2 discusses some tools we use. In §3 we present the new mapping. §4 gives cryptographic applications. In §5 we show how to implement our mapping, and in §6 we present simulation results.

2 Preliminaries

In this section we provide some background on the algebraic tori used in cryptography. See [23, 24] for more details. Let $\mathbb{F}_q$ denote the finite field with q elements, let ϕ be Euler's totient function, and let $\Phi_n(x)$ be the n-th cyclotomic polynomial. Write $G_{q,n}$ for the subgroup of $\mathbb{F}_{q^n}^{\times}$ of order $\Phi_n(q)$. Let $\mathbb{A}^n$ denote n-dimensional affine space. Recall that the Möbius function $\mu(m)$ is 0 if m is not square-free, and is $(-1)^k$ if m is a product of k distinct primes.

2.1 Algebraic Tori

For any positive integer n one can define an algebraic torus T_n over $\mathbb{F}_q$ whose $\mathbb{F}_q$-points consist of the elements of $\mathbb{F}_{q^n}^{\times}$ whose norm is one down to every proper subfield of $\mathbb{F}_{q^n}/\mathbb{F}_q$. The following provides some useful properties of T_n (see Lemma 7 of [23] and Lemma 1 of [2]).

Lemma 1. *1. $T_n(\mathbb{F}_q) \cong G_{q,n}$, and thus $\#T_n(\mathbb{F}_q) = \Phi_n(q)$.*
2. If $h \in T_n(\mathbb{F}_q)$ has prime order not dividing n, then $h \notin \mathbb{F}_{q^d}$ for any $d \mid n$ with $d < n$.

The variety T_n has dimension $\phi(n)$. Since $T_n(\mathbb{F}_q)$ embeds into $\mathbb{F}_{q^n}^{\times}$, one can perform the group operation as ordinary multiplication in the field, or use other more efficient possibilities [9]. The subgroup $T_n(\mathbb{F}_q)$ may be regarded as the "primitive" subgroup of $\mathbb{F}_{q^n}^{\times}$, since by Lemma 1 its elements do not lie in a proper subfield of $\mathbb{F}_{q^n}$. Thus, $T_n(\mathbb{F}_q)$ is believed to be the most cryptographically secure subgroup of $\mathbb{F}_{q^n}^{\times}$.

2.2 Rationality of Tori over $\mathbb{F}_q$

The interpretation of $G_{q,n}$ as an algebraic torus is motivated by the possibility, for some n, to exploit birational maps between T_n and affine space [23].

Definition 1. *The torus T_n is* **rational** *if there is a birational map $\rho : T_n(\mathbb{F}_q) \to \mathbb{A}^{\phi(n)}$.*

That is, $T_n(\mathbb{F}_q)$ is rational if there are Zariski open subsets $W \subset T_n$ and $U \subset \mathbb{A}^{\phi(n)}$, and rational functions $\rho_1, \ldots, \rho_{\phi(n)} \in \mathbb{F}_q(x_1, \ldots, x_n)$ and $\psi_1, \ldots, \psi_n \in \mathbb{F}_q(y_1, \ldots, y_{\phi(n)})$, such that $\rho = (\rho_1, \ldots, \rho_{\phi(n)}) : W \to U$ and $\psi = (\psi_1, \ldots, \psi_n) : U \to W$ are inverse isomorphisms (as quasi-projective varieties).

The existence of a birational map allows one to represent elements of $T_n(\mathbb{F}_q)$ with just $\phi(n)$ elements of $\mathbb{F}_q$, providing an effective compression factor of $n/\phi(n)$ over the embedding into $\mathbb{F}_{q^n}$. The torus T_n is known to be rational when n is either a prime power or a product of two prime powers [30, 12], and is conjectured to be rational for all n [30].

2.3 CEILIDH

The torus-based public key system CEILIDH was introduced at Crypto 2003 by Rubin and Silverberg [23]. The system is based on the rational torus T_6, and achieves a compression factor of three. They construct an efficiently computable bijection

$$\psi : \mathbb{A}^2(\mathbb{F}_q) \setminus V(f) \to T_6(\mathbb{F}_q) \setminus \{1, a\},$$

together with an efficiently computable inverse

$$\rho : T_6(\mathbb{F}_q) \setminus \{1, a\} \to \mathbb{A}^2(\mathbb{F}_q) \setminus V(f),$$

where $V(f)$ denotes a small subvariety of $\mathbb{A}^2$, and 1 and a in $T_6(\mathbb{F}_q)$ are points excluded by ρ and ψ. This allows one to represent all (bar two) elements of $T_6(\mathbb{F}_q)$ with just two elements of $\mathbb{F}_q$. This system attains the same compression factor as the public key system XTR [15, 29]. Granger, Page and Stam [9] give a comparison of XTR and CEILIDH in the case $q \equiv 2 \bmod 9$ or $q \equiv 5 \bmod 9$.

2.4 Asymptotically Optimal Torus-Based Cryptography

Since T_n is known to be rational only for special values of n, the above ideas do not lead to an optimal compression factor of $n/\phi(n)$ in general. Van Dijk and Woodruff [6] overcome this problem in the case where several elements of T_n are to be compressed. They construct a bijection:

$$\theta : T_n(\mathbb{F}_q) \times (\times_{d|n, \mu(n/d)=-1} \mathbb{F}_{q^d}^{\times}) \to \times_{d|n, \mu(n/d)=1} \mathbb{F}_{q^d}^{\times}. \tag{1}$$

Specializing their map to the case $n = 30$ gives

$$T_{30}(\mathbb{F}_q) \times \mathbb{F}_q^{\times} \times \mathbb{F}_{q^6}^{\times} \times \mathbb{F}_{q^{10}}^{\times} \times \mathbb{F}_{q^{15}}^{\times} \to \mathbb{F}_{q^2}^{\times} \times \mathbb{F}_{q^3}^{\times} \times \mathbb{F}_{q^5}^{\times} \times \mathbb{F}_{q^{30}}^{\times},$$

which can be reinterpreted as an "almost bijection" (see [6])

$$T_{30}(\mathbb{F}_q) \times \mathbb{A}^{32}(\mathbb{F}_q) \to \mathbb{A}^{40}(\mathbb{F}_q).$$

One can use this map to achieve an asymptotic compression factor of 30/8. Indeed, to compress m elements of $T_{30}(\mathbb{F}_q)$, one can compress an element x and split its image into $y_1 \in \mathbb{A}^8(\mathbb{F}_q)$ and $y_2 \in \mathbb{A}^{32}(\mathbb{F}_q)$. Then y_1 forms the affine input of the next compression. In the end, $8m+32$ elements of $\mathbb{F}_q$ are used to represent m elements of $T_{30}(\mathbb{F}_q)$. Observe that their map comes from the equation

$$\Phi_{30}(x)(x-1)(x^6-1)(x^{10}-1)(x^{15}-1) = (x^2-1)(x^3-1)(x^5-1)(x^{30}-1), \quad (2)$$

relating the orders of all the different component groups of domain and range. Since these groups are cyclic, one can map to and from their products as long as the orders of the component groups are coprime. For the map above there are some small primes that occur in the order of several component groups, but van Dijk and Woodruff are able to isolate and handle them separately.

3 The New Construction

The bijection (1), while asymptotically optimal, leaves open the question of whether one can obtain better compression for a *fixed* number of elements. Our new compression map, given by (4) below (see Theorems 2 and 4), has this property. Using the fact that $\Phi_n(x) = \prod_{d|n}(x^d-1)^{\mu(n/d)}$, we have

Proposition 1. If p is a prime, and a is a positive integer not divisible by p, then

$$\Phi_{ap}(x)\Phi_a(x) = \Phi_a(x^p).$$

The following result can be deduced from Proposition 1 above, using Lemma 6 of [6] (see also pp. 60–61 of [30]). Here, Res denotes the Weil restriction of scalars (see for example [30] or [24]).

Theorem 2. If p is a prime, q is a prime power, a is a positive integer, qa is not divisible by p, and $\gcd(\Phi_{ap}(q), \Phi_a(q)) = 1$, then

$$T_{ap}(\mathbb{F}_q) \times T_a(\mathbb{F}_q) \cong (\mathrm{Res}_{\mathbb{F}_{q^p}/\mathbb{F}_q} T_a)(\mathbb{F}_q) \cong T_a(\mathbb{F}_{q^p}).$$

The next result follows from Proposition 1, by doing double induction on the number of prime divisors of n and the number of prime divisors of m.

Theorem 3. If n is square-free and m is a divisor of n, then

$$\Phi_n(x) \prod_{d|\frac{n}{m},\, \mu(\frac{n}{md})=-1} \Phi_m(x^d) = \prod_{d|\frac{n}{m},\, \mu(\frac{n}{md})=1} \Phi_m(x^d).$$

The next result follows from Theorem 3, using the ideas in the proof of Theorem 3 of [6].

Theorem 4. If n is square-free and m is a divisor of n, then there is an efficiently computable bijection (with an efficiently computable inverse)

$$T_n(\mathbb{F}_q) \times \prod_{d|\frac{n}{m},\, \mu(\frac{n}{md})=-1} T_m(\mathbb{F}_{q^d}) \to \prod_{d|\frac{n}{m},\, \mu(\frac{n}{md})=1} T_m(\mathbb{F}_{q^d}).$$

Note that [6] is based on the case $m = 1$ of Theorem 4. Theorem 4 is most useful to us when T_m is rational. If T_m is rational, then Theorem 4 gives efficiently computable "almost bijections" between T_m and $\mathbb{A}^{\phi(m)}$, and we have

$$T_n \times \mathbb{A}^{D(m,n)} \sim \mathbb{A}^{\phi(n)+D(m,n)} \tag{3}$$

where

$$D(m,n) = \phi(m) \sum_{d|\frac{n}{m},\, \mu(\frac{n}{md})=-1} d$$

and $\sim$ denotes efficient "almost bijections". The smaller $D(m,n)$ is, the better for our applications. Given the current state of knowledge about the rationality of the tori T_m, we take m with at most two prime factors. Ideally, $m = 6$. One could also take $m = 2$. When $m = 6$, then (3) gives

$$T_{30} \times \mathbb{A}^2 \sim \mathbb{A}^{10} \qquad \text{and} \qquad T_{210} \times \mathbb{A}^{24} \sim \mathbb{A}^{72}.$$

As a comparison with the original bijection (1) for $n = 30$ which requires $8m+32$ elements of $\mathbb{F}_q$ to represent m elements in $T_{30}(\mathbb{F}_q)$, we see that this provides a considerable improvement.

Even better, using Proposition 1 and induction on the number of prime divisors of n, we also obtain the following.

Theorem 5. If $n = p_1 \cdots p_k$ is a product of $k \geq 2$ distinct primes, then

$$\Phi_n(x) \prod_{i=2}^{k-1} \Phi_{p_1 \cdots p_i}(x^{p_{i+2} \cdots p_k}) = \Phi_{p_1 p_2}(x^{p_3 \cdots p_k}).$$

Applying this to $n = 210 = 2 \cdot 3 \cdot 5 \cdot 7$, one can similarly show

$$T_{210}(\mathbb{F}_q) \times T_{30}(\mathbb{F}_q) \times T_6(\mathbb{F}_{q^7}) \sim T_6(\mathbb{F}_{q^{35}}).$$

Now since $T_6 \sim \mathbb{A}^2$, we obtain $T_{210} \times T_{30} \times \mathbb{A}^{14} \sim \mathbb{A}^{70}$. Using $T_{30} \times \mathbb{A}^2 \sim \mathbb{A}^{10}$ now gives $T_{210} \times \mathbb{A}^{22} \sim T_{210} \times \mathbb{A}^{10} \times \mathbb{A}^{12} \sim T_{210} \times (T_{30} \times \mathbb{A}^2) \times \mathbb{A}^{12} \sim T_{210} \times T_{30} \times \mathbb{A}^{14} \sim \mathbb{A}^{70}$, so

$$T_{210} \times \mathbb{A}^{22} \sim \mathbb{A}^{70}.$$

More generally, the above reasoning shows that if $n = p_1 \cdots p_k$ (square-free), then

$$T_n \times \mathbb{A}^{\phi(p_1 p_2) p_3 \cdots p_n - \phi(n)} \sim \mathbb{A}^{\phi(p_1 p_2) p_3 \cdots p_n},$$

which for $6 \mid n$ gives

$$T_n \times \mathbb{A}^{n/3 - \phi(n)} \sim \mathbb{A}^{n/3}. \tag{4}$$

Using (4), one can compress m elements of $T_n(\mathbb{F}_q)$ down to just $(m-1)\phi(n)+n/3$ elements of $\mathbb{F}_q$ by either sequential or tree-based chaining as explained in §4.

3.1 Applying the Construction to T_{30}

Henceforth we focus on $n = 30$ since this improves upon previous schemes, has a straightforward parameter generation (see §5), and will be computationally efficient. Note that $\gcd(\Phi_{30}(q), \Phi_6(q)) = 1$. Indeed, using the first paragraph of the proof of Lemma 6 of [6], the only possible prime dividing $\gcd(\Phi_{30}(q), \Phi_6(q))$ is 5, but it is easy to see that regardless of q we have $\Phi_6(q) \bmod 5 \in \{1, 2, 3\}$, which proves our claim. By Theorem 2 we now have

$$T_{30}(\mathbb{F}_q) \times T_6(\mathbb{F}_q) \cong T_6(\mathbb{F}_{q^5}).$$

The compression is based on a sequence of maps

$$T_{30}(\mathbb{F}_q) \times (\mathbb{A}^2(\mathbb{F}_q) \setminus V(f)) \to T_{30}(\mathbb{F}_q) \times T_6(\mathbb{F}_q) \to T_6(\mathbb{F}_{q^5}) \to \mathbb{A}^2(\mathbb{F}_{q^5}) \setminus V(f_5),$$

where $V(f_5)$ denotes $V(f)$ over $\mathbb{F}_{q^5}$. We denote by θ the forward composition of the three maps above, and by θ^{-1} the composition of the inverses. Note that if we have m elements in $T_{30}(\mathbb{F}_q)$, we compress them down to $8m + 2$ elements of $\mathbb{F}_q$. Thus the compression outperforms CEILIDH and XTR when as few as two elements are compressed.

The first and last maps are based on CEILIDH decompression and compression, respectively. We discuss some possibilities for the map $\sigma(\cdot, \cdot)$ between $T_{30}(\mathbb{F}_q) \times T_6(\mathbb{F}_q)$ and $T_6(\mathbb{F}_{q^5})$ in §5 below.

3.2 Missing Points

With regard to the functionality of θ, the only remaining issue is that the outer two maps based on CEILIDH do not give everywhere-defined injections.

We can slightly modify the CEILIDH maps, so that for compression we get an injection $\psi' : \mathbb{A}^2(\mathbb{F}_q) \to T_6(\mathbb{F}_q) \times \{0, 1\}$ and for decompression an injection $\rho' : T_6(\mathbb{F}_q) \times \{0, 1\} \to \mathbb{A}^2(\mathbb{F}_q)$. Note that ψ' and ρ' need not be inverses. The two missing points in ρ's domain can easily be added by using a table lookup into two arbitrarily chosen points in $V(f)$. The resulting map is ρ'.

Given the different cardinalities of $T_6(\mathbb{F}_p)$ (namely $p^2 - p + 1$) and $\mathbb{A}^2(\mathbb{F}_p)$ (namely p^2), there are certain points in $\mathbb{A}^2(\mathbb{F}_p)$ that do not decompress. If we concentrate on the case $p \equiv 2 \bmod 9$ or $p \equiv 5 \bmod 9$, then the variety $V(f)$ is defined by $f(v_1, v_2) = 1 - v_1^2 - v_2^2 + v_1 v_2$. For fixed v_2 this has at most 2 roots, and if this is the case then their difference is $(4 - 3v_2^2)^{1/2}$. If this expression equals 2 then $v_2 = 0$, in which case $v_1 \in \{-1, 1\}$. Thus we have a map $\psi' : \mathbb{A}^2(\mathbb{F}_q) \to T_6(\mathbb{F}_q) \times \{0, 1\}$:

- If $f(v_1, v_2) \neq 0$, then $\psi'(v_1, v_2) = (\psi(v_1, v_2), 0)$,
- Else if $v_2 \neq 0$, then $\psi'(v_1, v_2) = (\psi(v_1 + 2, v_2), 1)$,
- Else $\psi'(v_1, v_2) = (\psi(v_1 + 1, v_2), 1)$,

where the extra bit indicates whether the input landed in the variety.

4 Applications

Our new map saves a significant amount of communication in applications where many group elements are transmitted. For instance the compression can be used to agree on a sequence of keys using Diffie-Hellman as in §5.1 of [6]. Other applications include verifiable secret sharing, electronic voting and mix-nets, and private information retrieval.

In our applications we compress many elements. This is done by using part of the output of the i-th element as the affine input for the compression of the $(i+1)$-st element. This sequential chaining is simple, but has the drawback of needing to decompress all elements in order to obtain the first element. Alternatively, one can use trees to allocate the output of previous compressions. For instance, the output of the first compression is split into five pieces, which are subsequently used as the affine input when compressing elements two through six. The output of the second compression is used to compress elements seven through twelve, etc. When compressing m elements, decompressing a specific element now takes $O(\log m)$ atomic decompressions on average.

4.1 ElGamal Encryption

Our first application is ElGamal encryption with a small message domain, where we obtain an additional 10% compression over CEILIDH even for the encryption of a single message. This contrasts starkly with the original mapping of [6] that cannot be used to achieve any savings for single-message encryption.

Let q and l be primes such that $l \mid \Phi_{30}(q)$. Let $g \in \mathbb{F}_{q^{30}}^{\times}$ have order l, so that $\langle g \rangle \subseteq T_{30}(\mathbb{F}_q)$. For random a, $1 \leq a \leq l-1$, let a be Bob's private key and $A = g^a$ his public key. Without loss of generality, let $\mathcal{M} = \{0, 1, \ldots, m-1\}$ be the set of possible messages. We assume that m, the cardinality of $\mathcal{M}$, is small. We apply the mapping of §3 to the generalized ElGamal encryption scheme.

Encryption (M):

1. Alice represents the message M as $g^M \in \langle g \rangle$.
2. Alice selects a random integer k, $1 \leq k \leq l$, and computes $d = g^k$.
3. Alice sets $e = g^M \cdot (g^a)^k$.
4. Alice expresses $d \in T_6(\mathbb{F}_{q^5})$ as $(d_1, d_2) \in \mathbb{A}^8(\mathbb{F}_q) \times \mathbb{A}^2(\mathbb{F}_q) \cong \mathbb{A}^2(\mathbb{F}_{q^5})$ by using CEILIDH. Alice compresses $e \in T_{30}(\mathbb{F}_q)$ and $d_2 \in \mathbb{A}^2(\mathbb{F}_q)$ as $\theta(e, d_2) = T$, and outputs (d_1, T).

Decryption (d_1, T):

1. Bob computes $\theta^{-1}(T) = (e, d_2)$ and uses CEILIDH to reconstruct d.
2. Bob uses his private key a to recover $g^M = d^{-a}e$.
3. Bob recovers M from g^M using the fact that M comes from a small domain (e.g., using Pollard lambda or a table lookup).

The ciphertext is represented as 18 symbols in $\mathbb{F}_q$, which is a 10% improvement over a solution in which CEILIDH is used to compress both d and e. Note that the mapping of [6] in §2.4 cannot be used to achieve any savings in this case.

Our scheme preserves homomorphy, that is, without knowing the secret key a one can compute the encryption of $M_1 + M_2$ given encryptions of M_1 and M_2 separately. This is useful in applications such as the efficient two-party computations proposed by Schoenmakers and Tuyls [26] which use homomorphic ElGamal encryption for a small number of messages.

Exactly as for XTR and CEILIDH (with 6 replaced by 30), the security of our schemes follows from the difficulty of the DDH problem in $\mathbb{F}_{q^{30}}^{\times}$, the fact that $T_{30}(\mathbb{F}_q)$ is the primitive subgroup of $\mathbb{F}_{q^{30}}^{\times}$, and the fact that our map and its inverse are efficiently computable.

The representation of M as g^M in $\langle g \rangle$ is not efficient when m is large. We leave it as an open problem to adapt our scheme to handle a larger message domain. We note that one solution is to use hybrid ElGamal encryption instead. Indeed, we may apply the mapping of §3 to hybrid ElGamal encryption, adapting a protocol in §5.3 of [6]. In general, though, this solution does not preserve the homomorphic property.

4.2 ElGamal Signatures

We apply the mapping of §3 to the generalized ElGamal signature scheme, adapting a protocol in §5.2 of [6]. Here the signature has the form (d, e), where $d \in \langle g \rangle$ and $1 \leq e \leq l-1$. The idea is to use part of e in the affine component of θ, which can be done without any loss since $\log_2 e \approx 160$ while $2 \log_2 q \approx 70$; see §5.5 for a discussion of parameters. Since the affine component of [6] is much larger, this is not possible in their setting.

For a random a, $1 \leq a \leq l-1$, let a be Alice's private key and $A = g^a$ her public key. Let $h : \{0,1\}^* \to \mathbb{Z}_l$ be a cryptographic hash function. We have the following generalized ElGamal signature scheme for input message M:

Signature Generation (M):

1. Alice selects a random secret integer k, $1 \leq k \leq l$, and computes $d = g^k$.
2. Alice then computes $e = k^{-1}(h(M) - ah(d)) \bmod l$.
3. Alice expresses e as $(e_1, e_2) \in \mathbb{F}_q^2 \times \{0,1\}^*$, computes $\theta(d, e_1) = T$, and outputs (e_2, M, T) as her signature.

Signature Verification (e_2, M, T):

1. Bob computes $\theta^{-1}(T) = (d, e_1)$ and recovers e.
2. Bob accepts the signature if and only if $A^{h(d)} d^e = g^{h(M)}$.

We note that in practice one has the alternative of using Schnorr's signature scheme, which already achieves optimal compression.

4.3 Voting Schemes

We will discuss a recent voting scheme by Kiayias and Yung [11], which is based on the discrete logarithm problem and for which we propose to use T_{30}. We give

a comparison with a cutting edge scheme based on Paillier encryption due to Damgård and Jurik [5].

Let L denote the number of voters. Each voter has a secret key a_i, and a public key g^{a_i}. The i-th voter chooses L random exponents $s_{i,j} \in \mathbb{Z}_l$ which satisfy $\sum_j s_{i,j} = 0$, where j ranges from 1 to L (the scheme is self-tallying, which basically means that the voters themselves serve as the talliers). Voter i computes and posts $g^{a_j s_{i,j}}$ for all j, and a zero-knowledge proof that his post is well-formed. Define $t_j = \sum_i s_{i,j}$ and observe that

(a) $\sum_j t_j = 0$,
(b) t_j is a random element in $\mathbb{Z}_l$ if at least one user is honest.

From the posts, the j-th voter can compute $g^{a_j t_j}$, and then by using a_j can also compute g^{t_j}. If f is another public generator of $\langle g \rangle$, the vote of the j-th voter is a bit b_j from which he can calculate and post $g^{t_j} f^{b_j}$. From all such posts, we have $\prod_j g^{t_j} f^{b_j} = f^{\sum_j b_j}$. Since the tally $\sum_j b_j \leq L$, it can be found with Pollard's lambda method in $O(\sqrt{L})$ multiplications (and one already needs $\Theta(L)$ multiplications to compute $\prod_j g^{t_j} f^{b_j}$).

Damgård and Jurik [5] propose a similar scheme as Kiayias and Yung, but use Paillier encryption [19] as a starting point. Again, there are L voters, L secret keys Sk_i and public keys Pk_i. The i-th voter chooses L random integers $s_{i,j}$ in a predefined range with $\sum_j s_{i,j} = 0$. Voter i posts $E_{Pk_j}(s_{i,j})$ for all j, and a zero-knowledge proof that these values are well-formed. Define t_j as above, and observe properties (a) and (b) again hold (the latter statistically). From the posts, voter j computes $E_{Pk_j}(t_j)$, and using Sk_j gets t_j. If his vote is b_j, he then posts $p_j = t_j + b_j$. Observe that $\sum_j p_j = \sum_j b_j$. Hence, tallying is more efficient than in the scheme of Kiayias and Yung, using L additions versus $O(L)$ multiplications.

Although the Paillier-based scheme can be expected to be faster, a scheme based on T_{30} is considerably more compact. We give an analysis of the communication required for both schemes under the assumption that one wants $\log_2 n = 1024$, and that one achieves the same level of security with $30 \log_2 q = 1024$ and a subgroup size of 160 bits.

The communication of the Kiayias-Yung scheme is dominated by the sending of $g^{a_j s_{i,j}}$ for all i, j, together with their zero-knowledge proofs. When looking at the zero-knowledge proofs used, one sees that each voter transmits $4L$ group elements and L exponents. Thus we can use T_{30} compression in $\mathbb{F}_{q^{30}}$ here. This results in roughly $4L \cdot 8 \log_2 q + 160L = 32L \log_2 q + 160L \approx 1253L$ bits per voter.

The communication of the Damgård-Jurik scheme is similarly dominated by the sending of the $E_{PK_j}(s_{i,j})$ for all i, j with their zero-knowledge proofs. We note that their encryption scheme E is not exactly the same as that of Paillier, but a modification of it where the ciphertext size is at least $3 \log_2 n$ bits, n being an RSA modulus. Moreover, each proof that $E_{PK_j}(s_{i,j})$ is well-formed costs at least $5 \log_2 n$ bits. Thus $8L \log_2 n = 8192L$ bits are transmitted per voter in this stage.

Hence our improvement is roughly a factor of 6.5. An additional optimization is for the bulletin board to compress the list of public keys g^{a_i} when distributing

this at the beginning. We note that although we improve in communication and bulletin board size, the computational workload has clearly increased.

4.4 Mix-Nets

A typical *re-encrypting* mix-net involves M servers that each process n ciphertexts. To process the batch, a server randomly permutes and rerandomises the ciphertexts. In the literature, both ElGamal and Paillier type schemes are used. We save on the communication for both sequential and parallel mix-nets [8]. In a *sequential re-encrypting mix-net*, server i processes the n ciphertexts, then passes them to server $i+1$, and after the M-th server is done, they perform some kind of threshold decryption. Here $n \cdot M$ bits are communicated using either Paillier or ElGamal, but we save using ElGamal together with T_{30} compression. Our savings is a factor of 30/8. In *parallel mixing*, the servers process disjoint batches of input ciphertexts in parallel, and in between processing rounds they transmit n/M ciphertexts to each other, and again we save using T_{30} compression.

5 Representations and Algorithms for T_{30}

In this section we discuss implementation issues concerning field representation, key generation, and efficient exponentiation.

5.1 Field Representations

Since $T_{30}(\mathbb{F}_q) \subset \mathbb{F}_{q^{30}}^{\times}$, we need a model of the latter that permits fast multiplication, squaring, inversion and a fast Frobenius automorphism. We also require arithmetic for T_6, over both $\mathbb{F}_q$ and $\mathbb{F}_{q^5}$. Since $T_{30}(\mathbb{F}_q) \subset T_6(\mathbb{F}_{q^5})$, we may model the arithmetic of $T_{30}(\mathbb{F}_q)$ by the latter, possibly at the risk of losing some optimizations.

The base field $\mathbb{F}_q$. We base our implementation on high performance arithmetic in $\mathbb{F}_q$ using the representational method of Montgomery [17, 3]. For T_{30} one is likely to use characteristics q between 32 and 64 bits long, corresponding to a 2-word value on a 32-bit architecture. We are careful to distinguish between those small, 2-word values required by $T_6(\mathbb{F}_{q^5})$ and more general values of q (which we need for comparison purposes). Essentially, we employ the trivial program specialisation techniques described by Avanzi [1] to construct compact, straight line code sequences for the 2-word case. This affords a significant improvement over code for general sizes of q. Other than the length, we do not rely on assumptions on the value of q, although one could expect incremental improvements by doing so. Also, our choice of extension degree poses some congruence conditions on q.

The extension $\mathbb{F}_{q^5}$. We use a degree five subfield of the degree 10 extension $\mathbb{F}_q[t]/(\Phi_{11}(t))$, and use the Gaussian normal basis $\{t+t^{10}, t^7+t^4, t^5+t^6, t^2+t^9, t^3+t^8\}$. For this to work we require that $q \neq \pm 1 \mod 11$ [18]. Since the extension degree is small, we perform inversions using the Itoh-Tsujii algorithm [10].

Table 1. Arithmetic costs

	$\mathbb{F}_{q^5}$	$T_6(\mathbb{F}_q)$	$T_6(\mathbb{F}_{q^5})$
Multiply	$15M + 75A$	$18M + 53A$	$270M + 1615A$
Square		$6M + 21A$	$90M + 555A$
Inverse	$65M + 300A + I$	$2A$	$10A$
Frobenius	0	$1A$	$5A$
Compress		$15M + 31A + I$	$290M + 1580A + I$
Decompress		$27M + 52A + I$	$470M + 2585A + I$

The torus T_6. For the torus T_6 we take $q \equiv 2 \mod 9$ or $q \equiv 5 \mod 9$ and use arithmetic based on the degree six extension field defined by adjoining a primitive ninth root of unity to the base field, as in [28, 23, 9]. Note that in T_6 we have virtually free inversion, as it is just the cube of the Frobenius automorphism.

5.2 Compression and Decompression

Our new compression and decompression algorithms require two components: CEILIDH and CRT. We use an implementation of CEILIDH as given in [9].

Although it seems that Chinese remaindering is straightforward, there is some flexibility in choosing the map $\sigma : T_{30}(\mathbb{F}_q) \times T_6(\mathbb{F}_q) \to T_6(\mathbb{F}_{q^5})$. Following [6] we have $\sigma(x, y) = x^\beta y^\alpha$, where $\alpha\Phi_{30}(q) + \beta\Phi_6(q) = 1$. The inverse is given by $\sigma^{-1}(z) = (z^{\Phi_6(q)}, z^{\Phi_{30}(q)})$. The cost of the forward computation (i.e., σ) is an exponentiation in $T_{30}(\mathbb{F}_q)$, an exponentiation in $T_6(\mathbb{F}_q)$, and a multiplication. Depending on the context, the exponentiation in $T_{30}(\mathbb{F}_q)$ may be combined with an exponentiation performed as part of a particular protocol. The inverse is a double exponentiation.

Also attractive is the simple $\sigma'(x, y) = xy$ with inverse $(\sigma')^{-1}(z) = (zy^{-1}, y)$ where $y = z^{\alpha\Phi_{30}(q)}$. Clearly the forward map only costs a multiplication. For the inverse we first compute y using a single exponentiation. Note that the exponent here is larger than in the case of σ, but the total amount of exponent is similar in both cases (although asymptotically it is not the total amount that counts, it is what is relevant in practice). Moreover, we are typically concerned with the case where the preimage $x \in T_{30}(\mathbb{F}_q)$ has an order l dividing $\Phi_{30}(q)$ so we know that z has order dividing $l(q^2 - q + 1)$, which we can use to reduce the exponent $\alpha\Phi_{30}(q)$. As noted before, the computation of y^{-1} is virtually free, so this method is clearly preferable to the first suggestion.

5.3 Arithmetic Costs

Let M, A, S and I represent the cost of multiplication, addition, squaring and inversion in $\mathbb{F}_q$, respectively. In Table 1 (cf. [9–Lemma 3]) we detail the relative costs for arithmetic in $\mathbb{F}_{q^5}$, and for both $T_6(\mathbb{F}_q)$ and $T_6(\mathbb{F}_{q^5})$. Compression and decompression are based on CEILIDH.

5.4 Exponentiation in T_{30}

In protocols, one is required to perform one of three operations involving exponentiation: a single exponentiation in $T_{30}(\mathbb{F}_q)$, a double exponentiation in $T_{30}(\mathbb{F}_q)$, or a single exponentiation in $T_6(\mathbb{F}_{q^5})$ (for the map $(\sigma')^{-1}$ described above). Since $T_{30}(\mathbb{F}_q) \subset T_6(\mathbb{F}_{q^5})$, we can perform all three of these in $T_6(\mathbb{F}_{q^5})$ using the methods developed in [28]; the main properties one can exploit are the degree two Frobenius automorphism and fast squaring.

In a subgroup of order l where $l|(q^{10} - q^5 + 1)$, we write an exponent m as $m \equiv m_1 + m_2 q^5 \mod l$, where m_1 and m_2 are approximately half the bit-length of l (as in [28]). One can find m_1 and m_2 very quickly having performed a one-time Gaussian two dimensional lattice basis reduction, and using this basis to find the closest vector to $(m, 0)^T$. Having split the exponent, to compute a^m for random a and m, we perform a double exponentiation $a^{m_1}(a^{q^5})^{m_2}$ using the Joint Sparse Form (JSF) of the integers m_1 and m_2 [27], which on average halves the number of pairs of non-zero bits in their paired binary expansion. The use of the JSF in the above is possible since we have virtually free inversion.

To perform a double exponentiation, we split both exponents as with the single exponentiation, and perform the necessary four-fold multi-exponentiation as a product of two double exponentiations, combining the required squarings.

In general one may also be able to exploit the additional structure of T_{30}, which possesses an automorphism of degree eight, namely, the Frobenius automorphism. One can in principle employ exactly the same method as above and perform an eight-fold multi-exponentiation. However for the parameter sizes we consider in this paper, we use a much simpler method based on the q-ary expansion of an exponent m. Specifically, since our value of q will be small we can write an exponent m as $m = \sum m_i q^i$.

For our implementation where q^{30} has size approximately 1024 bits, exponentiating by a 160 bit exponent consists of five terms in the q-ary expansion, and hence we perform a five-fold multi-exponentiation. To perform this one can use ideas of Proos [21], which extend the JSF to more than two exponents. However due to the amount of precomputation required, for exponents of cryptographic interest we use a naive combination of the JSF and the non-adjacent form (NAF). This results in an effective exponent length of around the same size as q, significantly reducing the number of squarings needed for exponentiation.

With regard to decompression, the exponent in this case is slightly longer than for a single exponentiation. Again we use a q-ary expansion, consisting of seven terms in this instance, and apply the JSF to three pairs of them and the NAF to the remaining one.

Note that for larger parameter choices, one can clearly construct more efficient multi-exponentiation methods than those we have optimised for 1024 bit fields. We omit the details.

Table 2. Parameter examples with 32-bit primes q

q	l
2229155309	93160782386666970926793003905767713282869775177 1
2527138379	96337326395931809093808923299783279122089990331 1
2559356147	92280031103738926188087357025158570257112159045 1
2925130259	89912218766668878045741706369171526797619851659 1
3020282723	73446353284644903154947818417059577590631818890 1
3734718203	78957213148679015685309335297789575772056697844 1

Table 3. Parameter examples with 64-bit primes q

q	l
9909125592335111369	10563848710885954232271151735680485281846211409030529108053 01
10640772970658245433	31701195851377774228329380147608510135042587235754310186428 71
11042402719715204339	11793457320856742836596216037177707357884093667661444666860 61
11391285666382073129	12936784122105485373205586989397273467867058847287060671336 51
11868436123416952031	12303522427960516917606437178097923937512251101056304951130 71
17174393702711641469	10706758786453699988488694552055524038697731542086350010017 21

5.5 Parameter Selection

Rubin and Silverberg discuss parameter selection for T_{30} in §3.10 of [25]. We followed their method, starting with primes $p \equiv 1 \bmod 30$ of about 30 (resp., 61) bits, finding 32-bit (resp., 64-bit) primes q of order 30 mod p such that $q \equiv 2 \bmod 9$ and $q \equiv 7 \bmod 11$, then using the Elliptic Curve Method to remove small prime factors from $\Phi_{30}(q)/p$, and checking to see if what remains is a prime of about 160 (resp., 200) bits. This results in parameters with q a 32-bit (resp., 64-bit) prime with the property that the order of $T_{30}(\mathbb{F}_q)$ is divisible by a 160-bit (resp., 200-bit) prime l. These choices give security equivalent to 960-bit (resp., 1920-bit) RSA security.

By suitably optimizing the Elliptic Curve Method parameter choices, we were able to generate parameters at a rate of about one example every minute or two for 32-bit primes q (with 160-bit primes l), using a Macintosh G5 dual 2.5GHz computer. For 64-bit primes q, we obtained examples with l a prime of between 198 and 202 bits at a rate of one every few hours. The parameters are like Diffie-Hellman parameters, in the sense that the same parameters can be used for all users, and for many applications do not need to be changed frequently. In Table 2 we list some examples with 32-bit primes q and 160-bit primes l. In Table 3 we list some examples with 64-bit primes q and 200-bit primes l.

6 Timings

In order to understand the real-world performance of our construction, we implemented the entire system and ran a number of timing experiments. Our main goal

Table 4. Timings of basic field and torus arithmetic

	$\mathbb{F}_{q_L}$	$\mathbb{F}_{q_S}$	$\mathbb{F}_{q_S{}^5}$	$T_6(\mathbb{F}_{q_L})$	$T_{30}(\mathbb{F}_{q_S})$
Addition	$0.80\mu s$	$0.52\mu s$	$0.82\mu s$		
Frobenius			$0.48\mu s$	$1.64\mu s$	$1.10\mu s$
Square	$2.51\mu s$	$0.61\mu s$		$13.80\mu s$	$21.61\mu s$
Multiply	$2.58\mu s$	$0.62\mu s$	$3.78\mu s$	$32.30\mu s$	$65.92\mu s$
Inverse	$92.71\mu s$	$2.04\mu s$	$16.03\mu s$	$1.82\mu s$	$1.29\mu s$

Table 5. Timings for exponentiations

	$T_6(\mathbb{F}_{q_L})$	$T_{30}(\mathbb{F}_{q_S})$
Compression		
Compress	$131.30\mu s$	$0.13ms$
Decompress	$188.61\mu s$	$4.92ms$
Exponentiation		
Binary	$5.21ms$	$9.12ms$
Sliding Window	$4.39ms$	$7.53ms$
q-ary		$3.11ms$
JSF Single	$2.79ms$	$4.57ms$

is to compare the performance of an implementation of $T_6(\mathbb{F}_{q_L})$ using CEILIDH against an implementation of $T_{30}(\mathbb{F}_{q_S})$ of similar cardinality using our construction. Here, we denote the special cases of large and small q as q_L and q_S. We used $\log_2(q_L) \approx 5 \cdot \log_2(q_S) \approx 176$ bits, so that in both cases, there is a subgroup of roughly $\log_2(l) \approx 160$ bits in size. These parameters heuristically provide the equivalent of 1024-bit RSA security.

We constructed our implementation entirely in C++, apart from small sequences of assembly language to accelerate arithmetic in $\mathbb{F}_q$, using the GCC 3.4.2 compiler suite. The timing experiments were carried out on a Linux based PC housing a 2.8 GHz Intel Pentium 4 processor and 1 GB of memory. We selected our system parameters as in §5.5. In all of our timing experiments we generated random operands and averaged the timings of many experiments to get a representative result. Note that exponents are reduced modulo l in all cases. Our sliding window had a maximum size of four.

Table 4 shows timings for basic field and torus arithmetic. Arithmetic in $\mathbb{F}_{q_L}$ is used in $T_6(\mathbb{F}_{q_L})$ and arithmetic in $\mathbb{F}_{q_S}$ and $\mathbb{F}_{q_S{}^5}$ is used in $T_{30}(\mathbb{F}_{q_S})$. One interesting feature of these results is the low cost ratio between inversion and multiplication in $\mathbb{F}_{q_S}$, which is potentially interesting to follow up. Table 5 details the cost of mapping between different representations (compress and decompress) and the cost of different exponentiation methods which might be used within an actual cryptosystem.

It is difficult to get an exact comparison with other work on ECC and XTR, partly because of differences in host processor and levels of optimisation used by different authors in producing benchmark timings. However, a comparison

with the highly optimised ECC results of Avanzi [1], for example, gives some insight. For similar levels of security, direct comparison shows an exponentiation in $T_{30}(\mathbb{F}_{q^8})$ is only around twice as costly as an ECC point multiplication; correcting for the difference in processors still means that $T_{30}(\mathbb{F}_{q^8})$ is at least competitive. The case of XTR is easier to compare against since we essentially use the same experimental platform as that given in Granger, Page and Stam [9]. It turns out that XTR is marginally faster. Solely from the point of view of performance, we conclude that our construction is a competitive alternative to existing cryptosystems.

7 Conclusions

We construct an efficient "almost bijection" between $T_{30}(\mathbb{F}_q) \times \mathbb{A}^2(\mathbb{F}_q)$ and $\mathbb{A}^{10}(\mathbb{F}_q)$ which achieves better compression than XTR and CEILIDH for the compression of as few as two group elements. We give several applications, and obtain ElGamal ciphertexts that are 10% smaller than in previous schemes. We also develop an efficient implementation, using a variety of techniques for reducing the computational requirements and obtaining a scheme much more practical than that in [6]. From experimental results we conclude that our construction is a competitive alternative to the best existing public key cryptosystems.

References

[1] R.M. Avanzi. Aspects of Hyperelliptic Curves over Large Prime Fields in Software Implementations. *CHES'04*, LNCS **3156**, 148–162.

[2] W. Bosma, J. Hutton and E. R. Verheul. Looking beyond XTR. *Asiacrypt'02*, LNCS **2501**, 46–63.

[3] A. Bosselaers, R. Govaerts and J. Vandewalle. Comparison of Three Modular Reduction Functions. *Crypto'94*, LNCS **773**, 175–186.

[4] A.E. Brouwer, R. Pellikaan and E.R. Verheul. Doing More with Fewer Bits. *Asiacrypt'99*, LNCS **1716**, 321-332.

[5] I. Damgard and M. Jurik. A Length-Flexible Threshold Cryptosystem with Applications *ACISP'03*, LNCS **2727**, 350–364.

[6] M. van Dijk and D. Woodruff. Asymptotically Optimal Communication for Torus-Based Cryptography. *Crypto'04*, LNCS **3152**, 157–178.

[7] S. Goldwasser and S. Micali. Probabilistic Encryption. In *Comp. Sys. Sci*, **28**(1), 270–299, 1984.

[8] P. Golle and A. Juels. Parallel Mixing. In *Computer and Communications Security (CSS)*, ACM, 220–226, 2004.

[9] R. Granger, D. Page, and M. Stam. A comparison of CEILIDH and XTR. *ANTS-VI*, LNCS **3076**, 235–249, 2004.

[10] T. Itoh and S. Tsujii. A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases. In *Info. and Comp.*, **78**(3), 171–177, 1988.

[11] A. Kiayias and M. Yung. Self-Tallying Elections and Perfect Ballot Secrecy. *PKC'02*, LNCS **2274**, 141–158.

[12] A.A. Klyachko. On the Rationality of Tori with Cyclic Splitting Field (Russian). In *Arithmetic and Geometry of Varieties*, Kuybyshev Univ. Press, 73–78, 1988.

[13] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Foundations of Computer Science (FOCS)*, IEEE Press, 364–373, 1997.
[14] M.J.J. Lennon and P.J. Smith. LUC: A New Public Key System. In *IFIP TC11 Ninth International Conference on Information Security IFIP/Sec*, 103–117, 1993.
[15] A.K. Lenstra and E.R. Verheul. The XTR Public Key System. *Crypto'00*, LNCS **1880**, 1–19.
[16] A.K. Lenstra and E.R. Verheul. An Overview of the XTR Public Key System. In *Public-Key Cryptography and Computational Number Theory*, Verlages Walter de Gruyter, 151–180, 2001.
[17] P.L. Montgomery. Modular Multiplication Without Trial Division. In *Math. Comp.*, **44**, 519–521, 1985.
[18] M. Nöcker. Data structures for parallel exponentiation in finite fields. PhD Thesis, Universität Paderborn, 2001.
[19] P. Paillier. Public-key cryptosystems based on composite degree residuaosity classes. *Eurocrypt'99*, LNCS **1592**, 223–238.
[20] S.C. Pohlig and M.E. Hellman. An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance. In *IEEE Trans. on IT*, **24**, 106–110, 1978.
[21] J. Proos. Joint Sparse Forms and Generating Zero Columns when Combing. University of Waterloo, Technical Report CORR 2003-23.
[22] K. Rubin and A. Silverberg. Supersingular Abelian Varieties in Cryptology. *Crypto'02*, LNCS **2442**, 336–353.
[23] K. Rubin and A. Silverberg. Torus-Based Cryptography. *Crypto'03*, LNCS **2729**, 349–365.
[24] K. Rubin and A. Silverberg. Algebraic Tori in Cryptography. In *High Primes and Misdemeanours: Lectures in Honour of the 60th birthday of Hugh Cowie Williams*, Fields Institute Communications Series 41, American Mathematical Society, 317–326, 2004.
[25] K. Rubin and A. Silverberg. Using Primitive Subgroups to Do More with Fewer Bits. *ANTS-VI*, LNCS **3076**, 18–41, 2004.
[26] B. Schoenmakers and P. Tuyls, Practical Two-Party Computation Based on the Conditional Gate, *Asiacrypt'04*, LNCS 3329, 119–136.
[27] J.A. Solinas. Low-Weight Binary Representations for Pairs of Integers. University of Waterloo, Technical Report CORR 2001-41.
[28] M. Stam and A.K. Lenstra. Efficient Subgroup Exponentiation in Quadratic and Sixth Degree Extensions. *CHES'02*, LNCS **2523**, 318–332.
[29] M. Stam and A.K. Lenstra. Speeding Up XTR. *Asiacrypt'01*, LNCS **2248**, 125–143.
[30] V.E. Voskresenskiĭ. Algebraic Groups and Their Birational Invariants. *Translations of Mathematical Monographs* 179, American Mathematical Society, 1998.
[31] A. Yamamura and T. Saito. Private Information Retrieval Based on the Subgroup Membership Problem. *ACISP'01*, LNCS **2119**, 206–220.

A Tool Kit for Finding Small Roots of Bivariate Polynomials over the Integers

Johannes Blömer and Alexander May

Faculty of Computer Science, Electrical Engineering and Mathematics,
University of Paderborn,
33102 Paderborn, Germany
{bloemer, alexx}@uni-paderborn.de

Abstract. We present a new and flexible formulation of Coppersmith's method for finding small solutions of bivariate polynomials $p(x, y)$ over the integers. Our approach allows to maximize the bound on the solutions of $p(x, y)$ in a purely combinatorial way. We give various construction rules for different shapes of $p(x, y)$'s Newton polygon. Our method has several applications. Most interestingly, we reduce the case of solving univariate polynomials $f(x)$ modulo some composite number N of unknown factorization to the case of solving bivariate polynomials over the integers. Hence, our approach unifies both methods given by Coppersmith at Eurocrypt 1996.

Keywords: Coppersmith's method, univariate vs. bivariate, RSA

1 Introduction

In 1996, Coppersmith [6, 7, 8, 9] introduced two rigorous lattice-based methods for finding small roots of polynomials: One for univariate modular and another one for bivariate integer polynomial equations. Additionally, Coppersmith proposed heuristic multivariate extensions for both approaches. The goal in both methods is to maximize the bounds up to which roots of the polynomials can be found in polynomial time. Coppersmith's method for finding small solutions of modular polynomial equations has been applied in many settings, mainly for cryptanalytic purposes [1, 3, 4, 11] but also for proving the security of schemes [2, 15].

In contrast, the method for finding roots of polynomial equations over the integers has not found so many applications, yet. The most well-known result is the so-called factoring with high bits known [7, 8]: Let $N = pq$ be an RSA modulus and suppose we are given half of the high-order bits of p, then N can be factored in polynomial time. Recently, May [13] gave another application for the bivariate method: He showed that if the RSA secret key is known, then N can be factored in deterministic polynomial time. However, both results can also be proven using univariate polynomial equations.

In 1997, Howgrave-Graham [10] gave an easily applicable reformulation of Coppersmith's univariate modular method. This might be one of the reasons

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 251–267, 2005.

that up to now the univariate modular approach has found more applications than the bivariate integer approach. At Eurocrypt '04, Coron [5] succeeded to give a similar reformulation of Coppersmith's method over the integers.

While it is clear how to optimize a lattice basis for a given univariate polynomial of fixed degree, the construction of an optimal lattice basis for a bivariate polynomial $p(x, y)$ depends on the monomials that appear in $p(x, y)$. Coppersmith [8] analyzed the cases where $p(x, y)$ either has degree δ in x and y separately or degree δ in total.

Let us define the Newton polygon of $p(x, y)$ as the convex hull of the point set

$$\{(i, j) \in \mathbb{N}^2 \mid \text{monomial } x^i y^j \text{ appears in } p(x, y) \text{ with non-zero coefficient}\}.$$

For $p(x, y)$ with degree δ in each variable separately, the shape of the Newton polygon is a square. For $p(x, y)$ with total degree δ, the shape is an equilateral lower triangle (having his right angle in the lower left corner). These two shapes were also analyzed by Coron [5]. In addition, Coppersmith [8] mentions the case where the maximal degree of $p(x, y)$ in x is δ_x and the maximal degree in y is δ_y, which corresponds to a rectangle with side lengths δ_x and δ_y.

In this work, we provide a method that can be used to analyze arbitrary shapes of the Newton polygon of $p(x, y)$. One advantage of our main result is that we can formulate it just in terms of the monomials of $p(x, y)$. Although the proof of our main result requires lattice-based techniques, using our theorem the analysis of different shapes of $p(x, y)$ is purely combinatorial and can be done without any lattice theory. Hence, one can view our approach as a tool kit: If we are given a polynomial $p(x, y)$, we can maximize the bounds up to which a solution can be found in polynomial time. More precisely, let X and Y be upper bounds on the desired roots of $p(x, y)$. I.e., we want to find all solutions (x_0, y_0) such that $p(x_0, y_0) = 0$ and $|x_0| \leq X$, $|y_0| \leq Y$. Our goal is to maximize X and Y. The formulation of our main theorem allows to specify this maximization problem as an optimization problem over two sets of monomials. No lattice theory is required and the theorem can be used as a black box for cryptanalysts.

The proof of our main theorem is a variation of Coppersmith's original proof for the bivariate method [8]. We could use Coron's approach [5] for the proof of our result as well, but we prefer Coppersmith's approach since it has a crucial advantage: We usually obtain bounds of the form $XY \leq W^{g(\delta)-\epsilon}$, where $g(\delta)$ is some function in the degree of $p(x, y)$ in x, y and $W = \|p(xX, yY)\|_\infty$ is the max-norm of the coefficient vector of $p(xX, yY)$. The running time of Coppersmith's algorithm is polynomial in $(\log W, \delta, \frac{1}{\epsilon})$, while Coron's approach is polynomial in $(\log W, \delta)$ but exponential in $\frac{1}{\epsilon}$. This difference is due to a clever trick of Coppersmith which significantly reduces the dimension of the lattice involved by considering only a certain sublattice.

As applications of our main result, we provide rules to analyze different shapes of a Newton polygon of $p(x, y)$, thereby deriving some of the most well-known cryptographic results of Coppersmith's method. Hence, one can also see our

new method as a unifying method for certain different approaches to find small roots of polynomial equations. In particular, we obtain the following results for different shapes of the Newton polygons:

Rectangle: The rectangle can be seen as a warm-up example. Let us define $W = \|p(xX, yY)\|_\infty$. For polynomials of degree δ in each variable separately, we show the Coppersmith bound [8]

$$XY \leq W^{\frac{2}{3\delta}-\epsilon}.$$

Lower triangle: We analyze $p(x, y)$ with variable degree in x and y. When the total degree of $p(x)$ is δ, we obtain Coppersmith's bound [8]

$$XY \leq W^{\frac{1}{\delta}-\epsilon}.$$

Moreover, let us consider a univariate modular polynomial equation $f(x) = 0 \bmod N$, where f has degree δ. This can also be written as a bivariate polynomial $p(x, y) = f(x) - yN$ over the integers. The shape of $p(x, y)$'s Newton polygon is also a lower triangle, but with side-lengths δ and 1.

Our analysis shows that one can find all roots (x_0, y_0) of $p(x, y)$ over the integers provided that

$$|x_0| \leq N^{\frac{1}{\delta}},$$

which is exactly Coppersmith's result for univariate modular equations [8]. This unifies both approaches of Coppersmith from Eurocrypt '96 [6, 7]: The univariate modular case is already included in the bivariate integer case.

Surprisingly, the lattice basis underlying this result does not use powers of the polynomial $p(x, y)$, whereas in the univariate modular case it seems necessary to use powers of $p(x)$ in order to achieve the bound $N^{\frac{1}{\delta}}$.

Upper triangle: To our knowledge, the shape of an upper triangle (where the right angle is in the upper right corner) has not been analyzed in the literature before.

We use this shape to analyze the factorization algorithm for RSA-moduli $N = p^r q$, $r \geq 1$ of Boneh, Durfee and Howgrave-Graham [4]. In the original work, this is done using a variant of Coppersmith's univariate approach, namely one works modulo the divisor p^r of N. Interestingly, one can solve equations modulo p^r although one knows only N. Boneh, Durfee and Howgrave-Graham propose to exhaustively search approximations $\tilde{p}$ of p. For each guess $\tilde{p}$, they try to solve the polynomial equation $(\tilde{p} + x)^r = 0 \bmod p^r$, which has the solution $p - \tilde{p}$.

Alternatively, for each guess $\tilde{p}$ we consider the bivariate polynomial $f(x, y) = (\tilde{p} + x)^r y - N$ with the solution $(x_0, y_0) = (p - \tilde{p}, q)$. Notice that the shape of $f(x, y)$'s Newton polygon is an upper triangle. Our analysis yields the same result as the one in the work of Boneh, Durfee and Howgrave-Graham: One can find the factorization of N provided that

$$|x_0| \leq N^{\frac{r}{(r+1)^2}}.$$

Surprisingly, for $r > 1$ the following approach gives a smaller bound: Compute $\tilde{q} = \frac{N}{\tilde{p}}$ and try to solve the polynomial $f'(x,y) = (\tilde{p}+x)^r(\tilde{q}+y) - N$. Let X, Y be upper bounds on the desired solution $(x_0, y_0) = (p-\tilde{p}, q-\tilde{q})$. At first glance, the polynomial $f'(x,y)$ seems to be superior since we can decrease the size of Y. On the other hand, $W = \|p(xX, yY)\|_\infty$ decreases as well and the shape of $f'(x,y)$'s Newton polygon now is a rectangle, which has an inferior analysis. These two facts together outweigh the benefit of decreasing Y and we obtain a smaller bound.

In the case $r = 1$, both approaches give the same bound $|x_0| = |p-\tilde{p}| \leq N^{\frac{1}{4}}$. But still, the first approach should be preferred in practice since it uses a smaller lattice basis. So counterintuitively, one should sometimes ignore information about one variable in order to obtain a better shape of the Newton polygon. As the moral of this story, one should keep in mind that optimizing Coppersmith's bivariate method is not only a matter of optimizing the bounds X, Y but also of optimizing the structure of the underlying polynomial $p(x,y)$ itself!

In addition to the results above, we also prove general bounds for univariate polynomials of degree δ modulo some divisor b of N. The bounds are functions of the sizes of δ, b and N.

Rectangle and lower triangle: As a last example, we show how to combine two basic shapes such that all results for rectangles and/or for lower triangles follow as special cases by parameter settings.

We expect that similar to Coppersmith's approach [8] our bivariate method extends to a heuristic method for general multivariate equations, but we have not checked this so far.

The paper is organized as follows: In Section 2, we give our main result that allows to formulate the maximization problem of X and Y as an optimization problem for sets of monomials. In Section 3, we formulate our construction rules for the different shapes of Newton polygons of $p(x,y)$. Applications of these shapes are given in Section 4.

2 The Main Theorem

In this section we state our main theorem. We also describe the general setting in which we are going to apply the theorem in the following sections. First we need a couple of preliminary remarks and definitions.

Let M be a set of monomials in the variables x, y. We say that a polynomial $g(x,y)$ is *defined over* M or is a *polynomial over* M iff $g(x,y)$ can be written as

$$g(x,y) = \sum_{\mu \in M} c_\mu \mu, c_\mu \in \mathbb{Z}.$$

The proof of our main result uses a certain resultant that is required to be non-zero. In order to prove this property, the following definition is going to be useful. Later we will elaborate on this definition.

Definition 1. *Let $p(x,y)$ be a bivariate integer polynomial and S, M be finite non-empty sets of monomials in the variables x, y. The sets S, M are called* admissible *for $p(x,y)$ iff*

1. *For every monomial $\alpha \in S$ the polynomial $\alpha \cdot p(x,y)$ is defined over M.*
2. *For every polynomial g defined over M, if $g(x,y) = f(x,y) \cdot p(x,y)$ for some polynomial f, then f is defined over S.*

We say that an integer polynomial $p(x,y) \in \mathbb{Z}[x,y]$ is *irreducible* if $p(x,y) = f(x,y) \cdot g(x,y)$ with $f(x,y), g(x,y) \in \mathbb{Z}[x,y]$ implies that either $f(x,y) = \pm 1$ or $g(x,y) = \pm 1$. In particular, the gcd of all coefficients of an irreducible polynomial $p(x,y)$ must be 1.

Using these definitions we can already state our main theorem. Its proof can be found in the full version of the paper.

Theorem 2. *Let $p(x,y) \in \mathbb{Z}[x,y]$ be an irreducible integer polynomial in two variables with degree at most $d_x, d_y \geq 1$ in the variables x and y, respectively. Let $X, Y \in \mathbb{N}$ and set $W := \|p(xX, yY)\|_\infty$. Furthermore let $S, M, S \subseteq M$, be admissible for $p(x,y)$. Set*

$$s := |S|, \qquad m := |M|$$

$$s_x := \sum_{x^i y^j \in M \setminus S} i, \qquad s_y := \sum_{x^i y^j \in M \setminus S} j.$$

All pairs $(x_0, y_0) \in \mathbb{Z}^2$ satisfying

$$p(x_0, y_0) = 0 \quad \text{with } |x_0| \leq X, |y_0| \leq Y$$

can be found in time polynomial in m, d_x, d_y and $\log(W)$ provided

$$X^{s_x} Y^{s_y} < W^s \cdot 2^{-(8+c)sd_xd_y}, \tag{1}$$

where we assume that $(m-s)^2 \leq csd_xd_y$ for some constant c.

In the following we call elements of the set S *shift monomials*. The set S itself will be called *the set of shift monomials*. Let us describe how we are going to apply Theorem 2. To do so, we will identify sets of monomials with sets in the Euclidean plane $\mathbb{R}^2$. More precisely, for a set A of monomials in two variables x, y we define $\{(i,j) \in \mathbb{N}^2 | x^i y^j \in A\}$ and the convex hull $\mathrm{conv}(\{(i,j) \in \mathbb{N}^2 | x^i y^j \in A\})$ of this set. To simplify the notation we call these sets A as well. It will always be clear from the context whether we talk about a set of monomials or about the corresponding sets in the plane. Next, for a polynomial $g(x,y) = \sum c_{ij} x^i y^j, c_{ij} \in \mathbb{R}$ we define a convex set $N(g)$ in the Euclidean plane, called the *Newton polygon* of g. We set

$$N(g) := \mathrm{conv}\{(i,j) \in \mathbb{N}^2 | c_{ij} \neq 0\}.$$

The Newton polygon of the polynomial $p(x,y) = 2 + y + 3xy$ is depicted in Fig. 1.

Now suppose we want to use Theorem 2 to determine roots of some polynomial $p(x,y)$. Of course, we want to choose the bounds X, Y as large as possible.

To do so, we need to choose sets S and M carefully under the constraint that S, M are admissible for $p(x, y)$. Once we have chosen S, there is an obvious choice for M in order to guarantee the first property in Definition 1. That is, we choose M as the set of monomials $x^i y^j$ such that (i, j) lies in the so-called *Minkowski sum* $N(p) + S$ of the Newton polygon $N(p)$ and S. Here the Minkowski sum $A + B$ of two sets A, B in R^2 is defined as

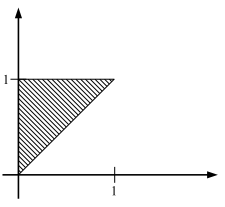

Fig. 1. Newton polygon of $2 + y + 3xy$

$$A+B := \{(a_1, a_2)+(b_1, b_2) \mid (a_1, a_2) \in A, (b_1, b_2) \in B\}.$$

As will be seen in our applications of Theorem 2, setting $M := N(p) + S$ will usually lead to a pair S, M of sets of monomials that also satisfies the second property of Definition 1, i.e. S, M will be admissible for the polynomial $p(x, y)$.

It remains to explain how to choose S in order to achieve large bounds X, Y, that satisfy Equation (1) in Theorem 2. Choosing good sets S requires a trade-off between the size s of S and the quantities s_x, s_y that depend on monomials in $M \setminus S$, where $M = N(p)+S$. We want s to be large, while s_x and s_y should stay relatively small. We have no provable method to find optimal sets S. However, the following general strategy proves to be successful.

We consider a whole class of sets S, that may be parametrized by several parameters. The shape of these sets resembles $N(p)$. Given these parametrized sets we determine the values s, s_x, s_y as functions of the parameters used to describe the sets. Finally, based on Equation (1) we determine the optimal setting for our parameters in order to get sets S, M and large bounds X, Y satisfying the conditions of Theorem 2.

3 The Constructions

Let us explain the construction of parametrized sets S for a few important shapes of Newton polygons $N(p)$ of polynomials $p(x, y)$. Applications of these examples and analysis of the bounds for X and Y that we can derive using these constructions will be given in the following section. First we define some important geometric shapes.

Definition 3. *In the following all parameters are real positive numbers.*

1. *Sets* $R(a, b) := \{x^i y^j \mid 0 \le j \le a, 0 \le i \le b\}$ *are called rectangles.*
2. *Sets* $L(c, a, \lambda) := \{x^{c+i} y^j \mid 0 \le j \le a, 0 \le i \le \lambda(a - j)\}$ *are called lower triangles.*
3. *Sets* $U(c, a, \lambda) := \{x^{c+i} y^j \mid 0 \le j \le a, 0 \le i \le \lambda j\}$ *are called upper triangles.*
4. *Sets* $E(c, a, \lambda) := R(a, c) \cup L(c, a, \lambda)$ *are called extended rectangles.*

Illustrations for these definitions are given in Fig. 2.
With these definitions we can state our main constructions.

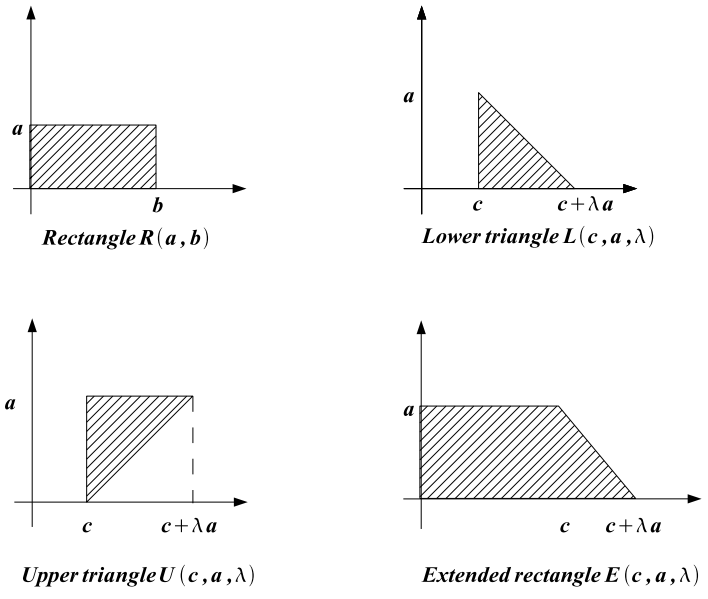

Fig. 2. Illustrations for Definition 3

Construction 4 (Rectangle construction). *Assume the Newton polygon $N(p)$ of polynomial $p(x, y)$ is the rectangle $R(d, \lambda d), \lambda > 0$. Then we use sets S such that*

$$x^i y^j \in S \Leftrightarrow (i, j) \in R(k, \gamma k).$$

Here $k \in \mathbb{N}$ and $\gamma > 0$. Consequently, the sets M of monomials are defined by

$$x^i y^j \in M \Leftrightarrow (i, j) \in R(k + d, \gamma k + \lambda d).$$

Furthermore

$$s = \sum_{j=0}^{k} \sum_{i=0}^{\gamma k} 1, \qquad m = \sum_{j=0}^{k+d} \sum_{i=0}^{\gamma k+\lambda d} 1$$

$$s_x = \sum_{j=0}^{k+d} \sum_{i=0}^{\gamma k+\lambda d} i - \sum_{j=0}^{k} \sum_{i=0}^{\gamma k} i, \qquad s_y = \sum_{j=0}^{k+d} \sum_{i=0}^{\gamma k+\lambda d} j - \sum_{j=0}^{k} \sum_{i=0}^{\gamma k} j.$$

In this construction the parameter γ is used to optimize the bounds X, Y.

In the rectangle construction as well as in the subsequent constructions, the parameter k is not used to optimize X, Y. Mainly it is used to control the size of certain low order error terms.

As it turns out the optimal γ is given by $\sqrt{\lambda}$, not by λ itself. Using the convex hulls of S and M instead of S, M itself, this construction is shown in Fig. 3.

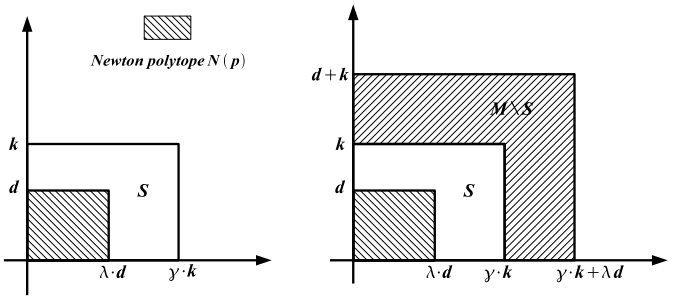

Fig. 3. The rectangle construction

Similarly, we define constructions for the lower and upper triangle, shown in Fig. 4. In the lower triangle construction we need no parameter to optimize the bounds X, Y.

Construction 5 (Lower triangle construction). *Assume the Newton polygon $N(p)$ of polynomial $p(x, y)$ is the lower triangle $L(0, d, \lambda), \lambda > 0$. Then we use sets S such that*

$$x^i y^j \in S \Leftrightarrow (i, j) \in L(0, k, \lambda).$$

Here $k \in \mathbb{N}$. Consequently, the sets M of monomials are defined by

$$x^i y^j \in M \Leftrightarrow (i, j) \in L(0, k + d, \lambda).$$

Using Definition 3, the formulas for s, m, s_x, and s_y can expressed in a similar fashion as in the rectangle construction.

Construction 6 (Upper triangle construction). *Assume the Newton polygon $N(p)$ of polynomial $p(x, y)$ is the upper triangle $U(0, d, \lambda), \lambda > 0$. Then we use sets S such that*

$$x^i y^j \in S \Leftrightarrow (i, j) \in R(k, ck) \cup U(ck, k, \lambda).$$

Here $k \in \mathbb{N}$ and $c \geq 0$. Consequently, the sets M of monomials are defined by

$$x^i y^j \in M \Leftrightarrow (i, j) \in R(k + d, ck) \cup U(ck, k + d, \lambda).$$

Again using Definition 3, the formulas for s, m, s_x, and s_y can expressed in a similar fashion as in the rectangle construction.

Of course, one can combine some or even all of these constructions into a single construction using several parameters to describe the shapes of $N(p)$ and S. For example, combining the rectangle and the lower triangle construction leads to the *extended rectangle construction*. This construction is shown in Fig. 5.

Our applications of Theorem 2 only use the constructions defined above. The following lemma shows that these constructions always yield admissible sets S and M. Hence in the subsequent sections we need not worry about the admissibility of the sets S and M that are used.

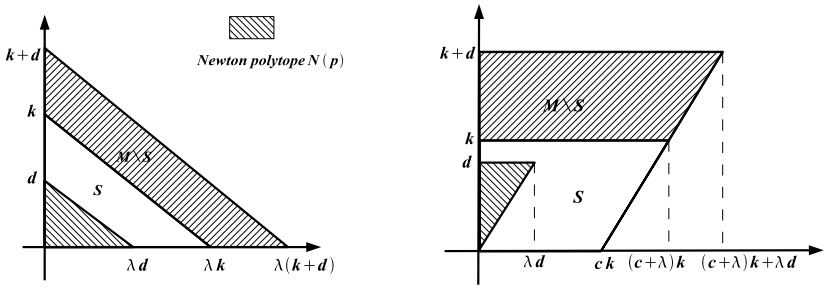

Fig. 4. Lower and upper triangle construction

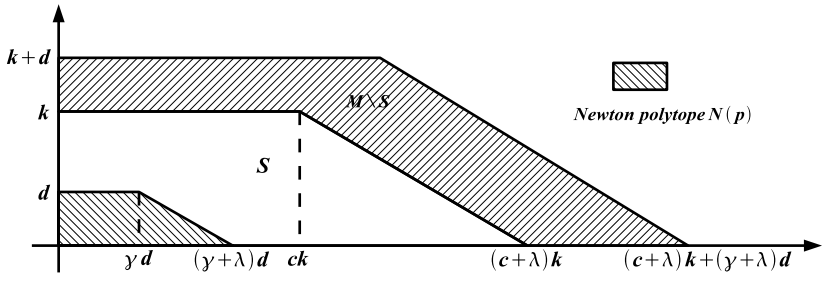

Fig. 5. The extended rectangle construction

Lemma 7. *The rectangle, lower triangle, upper triangle, and extended rectangle constructions as defined above lead to admissible sets S and M for the respective polynomials.*

Proof: We only show the lemma for the rectangle construction. The proofs for the other constructions are similar. As mentioned above, since M is the Minkowski sum of $N(p)$ and S, the sets S, M have the first property of Definition 1. To see that S, M also have the second property, consider a polynomial $f(x,y) = \sum f_{ij}x^iy^j$ that is not defined over S. We need to show that $f(x,y) \cdot p(x,y)$ is not defined over M. By l_x, l_y denote the degree of f in x, y, respectively. Since $f(x,y)$ is not defined over S, we have that $l_x > \gamma k$ or $l_y > k$. Since the two cases are symmetric, we only consider the case that $l_y > k$.

Let g be maximal over all i with $f_{il_y} \neq 0$. Then the coefficient of $x^{i+\lambda d}y^{l_y+d}$ in $f(x,y) \cdot p(x,y)$ will be non-zero. Since $l_y > k$ we get $l_y + d > k + d$ and $x^{i+\lambda d}y^{l_y+d} \notin M$. Hence $f(x,y) \cdot p(x,y)$ is not defined over M. □

4 Applications of Our Method

The following lemma is due to Coppersmith [8]. It is often used in the subsequent proofs to remove small error terms from the bounds. Namely, whenever we have a bound of B for the size of our solution, we can enlarge this bound to cB by

doing some brute-force search. This search increases the time complexity also by a factor of c.

Lemma 8 (Coppersmith). *Let $p(x,y) \in \mathbb{Z}[x,y]$. Assume that we have an algorithm A that finds all pairs $(x_0,y_0) \in \mathbb{Z}^2$ satisfying*

$$p(x_0,y_0) = 0 \quad \text{with } |x_0 \cdot y_0| \leq B$$

in time complexity T. Then one can find all (x_0,y_0) satisfying

$$p(x_0,y_0) = 0 \quad \text{with } |x_0 \cdot y_0| \leq cB$$

in time complexity cT.

Proof: We split our interval $[-cB, cB]$ into c subintervals of the size $2B$ centered at some x_i. For each of the subintervals with center x_i, we apply algorithm A to the polynomial $p(x - x_i, y)$ and output the roots in this subinterval. □

By Lemma 8, whenever we derive a bound of $B2^{-\mathcal{O}(\delta)}$ in the following theorems, we can also derive a bound of B by increasing the time complexity by a factor polynomial in 2^δ.

4.1 Rectangular Shape

We start by analyzing the case, where $p(x,y)$ has degree δ in x and y seperately.

Theorem 9 (Coppersmith). *Let $p(x,y) \in \mathbb{Z}[x,y]$ be an irreducible polynomial of degree δ in each variable separately. Let $X, Y \in \mathbb{N}$ and define $W = \|p(xX, yY)\|_\infty$. Then we can find all pairs $(x_0,y_0) \in \mathbb{Z}^2$ satisfying*

$$p(x_0,y_0) = 0 \quad \text{with } |x_0| \leq X, |y_0| \leq Y$$

in time polynomial in $\log W$ and δ provided that

$$XY \leq W^{\frac{2}{3\delta}} 2^{-\mathcal{O}(\delta)}.$$

Proof: Since the Newton polygon of our polynomial $p(x,y)$ is a rectangle, we apply Construction 4. We use the parameter setting

$$k = \max\{\log W, \delta\},\ \gamma = 1 \text{ and } \lambda = 1.$$

According to Construction 4, we shift our polynomial $p(x,y)$ with all the monomials in $S = R(k,k)$. Let $M = R(k+\delta, k+\delta)$. By Lemma 7, the sets S and M are admissible for $p(x,y)$ and Theorem 2 is applicable.
Plugging our values of $\gamma = \lambda = 1$ in the formulas for s_x, s_y, s and m gives us

$$s_x = s_y = \frac{3\delta}{2}k^2\Big(1 + \mathcal{O}\Big(\frac{\delta}{k}\Big)\Big),\ s \geq k^2 \text{ and } s, m = \mathcal{O}(k^2).$$

Furthermore, we have $d_x = d_y = \delta$. One easily checks the condition $(m-s)^2 = \mathcal{O}(sd_xd_y)$ of Theorem 2. An application of Theorem 2 with the values of s_x, s_y, s, d_x and d_y leaves us with the condition

$$(XY)^{\frac{3\delta}{2}k^2(1+\mathcal{O}(\frac{\delta}{k}))} \leq W^{k^2}2^{-\mathcal{O}(k^2\delta^2)}$$

This implies the bound

$$XY \leq W^{\frac{k^2}{\frac{3\delta}{2}k^2(1+\mathcal{O}(\frac{\delta}{k}))}}2^{-\mathcal{O}(\delta)}.$$

Now we observe that for any x, we have $\frac{1}{1+x} \leq 1-x$. Therefore, we can bound the exponent of W by $\frac{2}{3\delta}(1-\mathcal{O}(\frac{\delta}{k}))$. This leads to the new condition

$$XY \leq W^{\frac{2}{3\delta}}W^{-\mathcal{O}(\frac{1}{k})}2^{-\mathcal{O}(\delta)}.$$

Since we chose $k \geq \log W$, our term $W^{\mathcal{O}(\frac{1}{k})}$ is of constant size. An application of Lemma 8 shows that we can omit this term by increasing the running time only by a constant factor. This concludes the proof of the theorem. □

4.2 Lower Triangular Shape

First, we state the case where $p(x,y)$ has total degree δ. The proof of the following theorem can be found in the full version of the paper.

Theorem 10 (Coppersmith). *Let $p(x,y) \in \mathbb{Z}[x,y]$ be an irreducible polynomial of total degree δ. Let $X, Y \in \mathbb{N}$ and define $W = \|p(xX, yY)\|_\infty$. Then we can find all pairs $(x_0, y_0) \in \mathbb{Z}^2$ satisfying*

$$p(x_0, y_0) = 0 \quad \text{with } |x_0| \leq X, |y_0| \leq Y$$

in time polynomial in $\log W$ and δ provided that

$$XY \leq W^{\frac{1}{\delta}}2^{-\mathcal{O}(\delta)}.$$

Next, let us analyze the case $p(x,y) = f(x) - yN$, where $f(x)$ is a univariate polynomial of degree δ. This is exactly the univariate modular case and the following result reduces Coppersmith's univariate modular method [6] to the bivariate integer method [7].

In order to state Theorem 11, we use the following notation: Let $a_1, a_2, \ldots, a_n \in \mathbb{Z}$. We denote by $\gcd(a_1, a_2, \ldots, a_n)$ the greatest integer that divides all $a_i, i = 1 \ldots n$.

Theorem 11 (Coppersmith). *Let N be a composite integer of unknown factorization. Let $f(x) = \sum f_i x^i \in \mathbb{Z}[x]$ be a polynomial of degree δ with $\gcd(f_1, f_2, \ldots, f_\delta, N) = 1$. Furthermore, let $X \in \mathbb{N}$. Then we can find all point $x_0 \in \mathbb{Z}$ satisfying*

$$f(x_0) = 0 \bmod N \quad \text{with } |x_0| \leq X$$

in time polynomial in $\log N$ and δ provided that

$$X \leq N^{\frac{1}{\delta}}.$$

Proof: We define the following bivariate polynomial

$$p(x,y) = f_N(x) - yN,$$

where $f_N(x) = f(x) \bmod N$. I.e., we reduce the coefficients of $f(x)$ modulo N. Notice that x_0 is a root of $f(x_0)$ modulo N iff $p(x,y)$ has the root (x_0, y_0) for some y_0 over the integers. Furthermore, $p(x,y)$ is irreducible. Since we reduced $f(x)$ by N, we can upper bound the size of y_0 by

$$|y_0| \leq \frac{|f_N(x_0)|}{N} \leq X^\delta + X^{\delta-1} + \cdots + X^0 \leq (\delta+1)X^\delta.$$

Let us define $Y = (\delta+1)X^\delta$. Then we obtain $W = \|f(xX, yY)\|_\infty = YN$.

The shape of the Newton polygon of $p(x,y)$ is a lower triangle. Therefore, we apply Construction 5. Here, we use the parameter setting

$$k = \max\{\log W, \delta\},\ d = 1 \text{ and } \lambda = \delta.$$

That means, we apply the shifts with the monomials in $S = L(0,k,\delta)$ to the polynomial $f(x,y)$. Let $M = L(0,k+\delta,\delta)$. By Lemma 7 the sets S and M are admissible for $p(x,y)$, and Theorem 2 is applicable.

Setting the values $d = 1$ and $\lambda = \delta$ in our formulas for s_x, s_y, s and m provides us with the bounds

$$s_x = \frac{\delta^2}{2}k^2\Big(1+\mathcal{O}\Big(\frac{1}{k}\Big)\Big),\ s_y = \frac{\delta}{2}k^2\Big(1+\mathcal{O}\Big(\frac{1}{k}\Big)\Big),\ s \geq \frac{\delta}{2}k^2 \text{ and } s, m = \mathcal{O}(\delta k^2).$$

Furthermore, we observe that $d_x = \delta$ and $d_y = 1$. One easily checks that our parameters satisfy the condition $(m-s)^2 = \mathcal{O}(sd_xd_y)$ of Theorem 2.
Using these values in combination with Theorem 2 leads to the condition

$$X^{\frac{\delta^2}{2}k^2(1+\mathcal{O}(\frac{1}{k}))}Y^{\frac{\delta}{2}k^2(1+\mathcal{O}(\frac{1}{k}))} \leq W^{\frac{\delta}{2}k^2}2^{-\mathcal{O}(\delta^2k^2)}$$

Since $W = YN$, we obtain

$$X^{\frac{\delta^2}{2}k^2(1+\mathcal{O}(\frac{1}{k}))} \leq N^{\frac{\delta}{2}k^2}Y^{-\mathcal{O}(\delta k)}2^{-\mathcal{O}(\delta^2k^2)}$$

Analogous to the reasoning in the proof of Theorem 9, this implies the bound

$$X \leq N^{\frac{1}{\delta}}N^{-\mathcal{O}(\frac{1}{\delta k})}Y^{-\mathcal{O}(\frac{1}{\delta k})}2^{-\mathcal{O}(1)}. \tag{2}$$

By our setting, we have $k \geq \log W$ which bounds the term $(NY)^{-\mathcal{O}(\frac{1}{\delta k})} = W^{-\mathcal{O}(\frac{1}{\delta k})}$ by a constant. An application of Lemma 8 shows that we can increase the bound in (2) to the desired bound $X \leq N^{\frac{1}{\delta}}$ by increasing the running time by a constant factor.

By Theorem 2, we know that the running time of our algorithm is polynomial in $\log W$ and δ. It remains to show that $\log W$ is also a polynomial in $\log N$ and δ. Since our condition in inequality (2) implies that $X \leq N^{\frac{1}{\delta}}$, we have $W = YN = (\delta+1)X^\delta N \leq (\delta+1)N^2$ or equivalently $\log W \leq \log(\delta+1) + 2\log N$. This concludes the proof of the theorem. □

4.3 Upper Triangular Shape

In this subsection, we analyze a variant of Coppersmith's univariate modular approach, where one solves polynomial equations modulo a divisor of N. We start by reproducing the Boneh, Durfee and Howgrave-Graham [4] lattice-based factoring for RSA-moduli $N = p^r q$, $r \geq 1$, which is a generalization of "factoring with high bits known" of Coppersmith [8].

Theorem 12 (BDH). *Let $N = p^r q$ be an RSA modulus, where p and q are primes of the same bit-size and $r \geq 1$ is an integer. Suppose we are given an approximation $\tilde{p}$ of p with*

$$| p - \tilde{p} | \leq N^{\frac{r}{(r+1)^2}}.$$

Then we can find the factorization of N in time polynomial in $\log N$ and r.

Proof: We define the polynomial

$$f(x, y) = (\tilde{p} + x)^r y - N.$$

with the root $(x_0, y_0) = (p - \tilde{p}, q)$. Let $X = N^{\frac{r}{(r+1)^2}}$, then by our assumption $|x_0| \leq X$. Now, let us also find an upper bound Y for the size of $y_0 = q$. Since p and q are of the same bit-size, we know that $p > \frac{q}{2}$. Therefore, we obtain $q = \frac{N}{p^r} < \frac{2^r N}{q^r}$ which gives us $q^{r+1} < 2^r N$. This yields the upper bound $q < 2N^{\frac{1}{r+1}}$. Thus, we set $Y = 2N^{\frac{1}{r+1}}$. Obviously, we have $W = \|f(xX, yY)\|_\infty \geq N$.

Since the structure of the Newton polygon of our polynomial $f(x, y)$ is an upper triangle, we apply Construction 6. Here we use the parameter setting

$$k = \max\{\log N, r\}, d = 1, \lambda = r \text{ and } c = 1.$$

Thus, we use the shifts of the polynomial $f(x, y)$ with all the monomials in $S = R(k, k) \cup U(k, k, r)$. Let $M = R(k+1, k) \cup U(k, k+1, r)$. By Lemma 7, the sets S and M are admissible. Therefore, Theorem 2 is applicable.
Plugging the values $d = 1, \lambda = r$ and $c = 1$ into our formulas for s_x, s_y, s and m yields

$$s_x = \tfrac{(r+1)^2}{2} k^2 \Big(1 + \mathcal{O}\Big(\tfrac{1}{k}\Big)\Big), s_y = \big(r+1\big)k^2 \Big(1 + \mathcal{O}\Big(\tfrac{1}{k}\Big)\Big)$$
$$s \geq \Big(\tfrac{r}{2} + 1\Big)k^2 \text{ and } s, m = \mathcal{O}(rk^2)$$

Furthermore, we have $d_x = r$ and $d_y = 1$. One can check that these parameters meet the condition $(m - s)^2 = \mathcal{O}(s d_x d_y)$ of Theorem 2.
Now we apply Theorem 2 with the above parameters, which gives us

$$X^{\frac{(r+1)^2}{2} k^2 (1 + \mathcal{O}(\frac{1}{k}))} Y^{(r+1)k^2 (1 + \mathcal{O}(\frac{1}{k}))} \leq W^{(\frac{r}{2}+1)k^2} 2^{-\mathcal{O}(r^2 k^2)}.$$

Using $Y = 2N^{\frac{1}{r+1}}$ and $W \geq N$ leads to the new condition

$$X^{\frac{(r+1)^2}{2} k^2 (1 + \mathcal{O}(\frac{1}{k}))} \leq N^{\frac{r}{2} k^2 - \mathcal{O}(k)} 2^{-\mathcal{O}(r^2 k^2)}.$$

This in turn gives us

$$X \leq N^{\frac{\frac{r}{2}k^2}{\frac{(r+1)^2}{2}k^2(1+\mathcal{O}(\frac{1}{k}))}} N^{-\mathcal{O}(\frac{1}{r^2k})} 2^{-\mathcal{O}(1)},$$

which can be transformed into

$$X \leq N^{\frac{r}{(r+1)^2}} N^{-\mathcal{O}(\frac{1}{rk})} 2^{-\mathcal{O}(1)}.$$

Since $k \geq \log N$, an application of Lemma 8 gives us the desired bound $X \leq N^{\frac{r}{(r+1)^2}}$ by an increase of the running time by a constant factor. □

For the special case $r = 1$, we use the polynomial $p(x, y) = (\tilde{p} + x)y - N$ in the analysis of the proof of Theorem 12. In contrast, Coppersmith [8] proposed to use the polynomial $p'(x, y) = (\tilde{p} + x)(\tilde{q} + y) - N$, where $\tilde{q} = \frac{N}{\tilde{p}}$.

For $r = 1$, both polynomials give the same bound (but $p(x, y)$ yields smaller lattice bases, so it should lead to a faster algorithm in practice). Interestingly, for $r > 1$ the polynomial $(\tilde{p} + x)^r y - N$ yields a better bound than its counter-part with $\tilde{q}$, although we have to increase the bound on y_0. But this disadvantage is outweighed by the fact that the shape of $p(x, y)$ is upper triangular rather than rectangular, and that we can increase W to N.

In the following theorem, we analyze the more general case where we want to solve a univariate polynomial $f(x)$ with $f(x_0) = \bar{c}b$ for some small root x_0 and some (unknown) divisor b of N. Here, we assume that $\bar{c}$ is a known constant. By the result of the theorem, a large $\bar{c}$ helps to improve the bound. Unfortunately, we are not aware of an application with $\bar{c} > 1$.

Theorem 13. *Let N be a composite integer of unknown factorization with divisor $b \geq N^\beta$. Let $f(x) = \sum f_i x^i \in \mathbb{Z}[x]$ be a polynomial of degree δ with $\gcd(f_1, f_2, \ldots, f_\delta, \bar{c}N) = 1$. Then we can find all points $x_0 \in \mathbb{Z}$ satisfying $f(x_0) = \bar{c}b$ for some known constant $\bar{c} = N^\gamma, \gamma \geq 0$ in time polynomial in $\log N, \delta$ and γ provided that*

$$|x_0| \leq N^{\frac{(\beta+\gamma)^2}{\delta(1+\gamma)}}.$$

Proof: We define the following bivariate polynomial

$$p(x, y) = f(x)y - \bar{c}N.$$

Notice that $p(x, \frac{N}{b})$ has the same roots as $f(x) - \bar{c}b$ over the integers. Furthermore, $p(x, y)$ is irreducible. Define $y_0 = \frac{N}{b}$. Since $b \geq N^\beta$, we know that $y_0 \leq N^{1-\beta}$. Let $Y = N^{1-\beta}$ denote this upper bound for y_0.

Next, we will determine all integer roots (x_0, y_0) of $p(x, y)$ with the property that $|x_0| \leq X$ and $|y_0| \leq Y$. Among these roots must be all roots of $f(x) - \bar{c}b$. (It may happen that we additionally find roots of $f(x) - \bar{c}b'$ for some other divisor b' of N.)

We observe that $W = \|f(xX, yY)\|_\infty \geq \bar{c}N$.

Notice that the structure of the Newton polygon of $p(x, y)$ is an upper triangle. Therefore, we apply Construction 6. In this case, we use the parameter setting

$$k = \max\{\log N, \delta, \gamma\},\ d = 1,\ \lambda = \delta \text{ and } c = \frac{(1-\beta)\delta}{\beta+\gamma}$$

That means that we shift the polynomial $p(x, y)$ with all the monomials in the set $S = R(k, ck) \cup U(ck, k, \delta)$. Let $M = R(k+1, ck) \cup U(ck, k+1, \delta)$. By Lemma 7 the sets S and M are admissible for $p(x, y)$. Therefore, Theorem 2 is applicable.

If we plug in the values of d, λ and c in our formulas for s_x, s_y, s and m, we obtain

$$s_x = \tfrac{\delta^2(1+\gamma)^2}{2(\beta+\gamma)^2}k^2\big(1+\mathcal{O}(\tfrac{1}{k})\big),\ s_y = \tfrac{\delta(1+\gamma)}{\beta+\gamma}k^2\big(1+\mathcal{O}(\tfrac{1}{k})\big),$$
$$s \geq \tfrac{\delta(2-\beta+\gamma)}{2(\beta+\gamma)}k^2 \text{ and } s, m = \mathcal{O}(\delta k^2)$$

Notice that $d_x = \delta$ and $d_y = 1$. We easily check that the condition $(m-s)^2 = \mathcal{O}(sd_xd_y)$ of Theorem 2 is satisfied.
Using $Y = N^{1-\beta}$ and $W \geq \bar{c}N = N^{1+\gamma}$, an application of Theorem 2 yields

$$X^{\frac{\delta^2(1+\gamma)^2}{2(\beta+\gamma)^2}k^2(1+\mathcal{O}(\frac{1}{k}))}N^{\frac{\delta(1+\gamma)(2-2\beta)}{2(\beta+\gamma)}k^2(1+\mathcal{O}(\frac{1}{k}))} \leq N^{\frac{\delta(1+\gamma)(2-\beta+\gamma)}{2(\beta+\gamma)}k^2}2^{-\mathcal{O}(\delta^2k^2)}.$$

This can be rewritten as

$$X^{\frac{\delta^2(1+\gamma)^2}{2(\beta+\gamma)^2}k^2(1+\mathcal{O}(\frac{1}{k}))} \leq N^{\left(\frac{\delta(1+\gamma)(2-\beta+\gamma)}{2(\beta+\gamma)}-\frac{\delta(1+\gamma)(2-2\beta)}{2(\beta+\gamma)}\right)k^2}N^{-\mathcal{O}(\delta k)}2^{-\mathcal{O}(\delta^2k^2)},$$

which simplifies to

$$X^{\frac{\delta^2(1+\gamma)^2}{2(\beta+\gamma)^2}k^2(1+\mathcal{O}(\frac{1}{k}))} \leq N^{\frac{\delta(1+\gamma)}{2}k^2}N^{-\mathcal{O}(\delta k)}2^{-\mathcal{O}(\delta^2k^2)}$$

This in turn gives us the new condition

$$X \leq N^{\frac{(\beta+\gamma)^2}{\delta(1+\gamma)}}N^{-\mathcal{O}(\frac{1}{\delta k})}2^{-\mathcal{O}(1)}$$

Since $k \geq \log N$, an application of Theorem 8 yields the desired bound. □

As the special case $\bar{c} = 1$ of Theorem 13, we obtain the following corollary.

Corollary 14. *Let N be a composite integer of unknown factorization with divisor $b \geq N^\beta$. Let $f(x) = \sum f_i x^i \in \mathbb{Z}[x]$ be a polynomial of degree δ with $\gcd(f_1, f_2, \ldots, f_\delta, N) = 1$. Then we can find all points $x_0 \in \mathbb{Z}$ satisfying $f(x_0) = b$ in time polynomial in $\log N$ and δ provided that*

$$|x_0| \leq N^{\frac{\beta^2}{\delta}}.$$

An application of Corollary 14 is again "factoring with high bits known" [8]: Let $N = pq$ with $p > q$. Define $f(x) = \tilde{p} + x$. We want to find $x_0 = p - \tilde{p}$ with $f(x_0) = p$. We have $p \geq N^{\frac{1}{2}}$, which implies $\beta = \frac{1}{2}$. Hence, we obtain the well-known bound $|x_0| \leq N^{\frac{1}{4}}$.

Another application is the deterministic reduction of May [13]: Let $N = pq$ be an RSA modulus and let (e, d) satisfy $ed = 1 \bmod \phi(N)$. Suppose, we are given (N, e, d). Define $f(x) = N - x$. We want to find $x_0 = p + q - 1 \approx N^{\frac{1}{2}}$ with $f(x_0) = \phi(N)$. Notice that we know the multiple $ed-1$ of $\phi(N)$. Let $ed-1 = N^{\alpha}$ with $\alpha \leq 2$. Then we can set $\beta = \frac{1}{\alpha}$. Therefore, we can recover x_0 as long as $|x_0| \leq N^{\frac{1}{\alpha}}$. Since $\alpha \leq 2$, our bound is at least of the desired size $N^{\frac{1}{2}}$.

Similar to the case of "factoring with high bits known", the reduction yields another polynomial than originally proposed by May. Here, we obtain the polynomial $p(x, y) = (N - x)y + 1 - ed$, whereas May suggested to use $p'(x, y) = (N - x)(\tilde{k} + y) + 1 - ed$ with $\tilde{k} = \frac{ed-1}{N}$. Again, we can ignore the knowledge provided by $\tilde{k}$ in the analysis without affecting the bound. As before, $p(x, y)$ should be preferred in practice since it yields smaller lattice bases.

We want to point out that a result similar to the bound given in Corollary 14 has been given by Howgrave-Graham [11]. He showed a bound of N^{β^2} for solving $f(x) = 0 \bmod b$, where $f(x)$ has degree 1. This was later generalized by May [14] to $N^{\frac{\beta^2}{\delta}}$ for $f(x)$ of degree δ. Notice that these approaches allow to solve $f(x) = c'b$ for some *unknown* c' as opposed to $f(x) = \bar{c}b$ for some *known* $\bar{c}$ as in Theorem 13.

We pose the open problem to reduce this case of unknown c' to the bivariate integer case or a provable trivariate integer case. To our knowledge, this is the only rigorous variant of Coppersmith's method which is not covered by our new approach.

References

1. J. Blömer, A. May, "New Partial Key Exposure Attacks on RSA", Advances in Cryptology – Crypto 2003, Lecture Notes in Computer Science Vol. 2729, pp. 27–43, Springer-Verlag, 2003
2. D. Boneh, "Simplified OAEP for the RSA and Rabin Functions", Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science Vol. 2139, pp. 275–291, Springer-Verlag, 2001
3. D. Boneh, G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$", IEEE Trans. on Information Theory, Vol. 46(4), pp. 1339–1349, 2000
4. D. Boneh, G. Durfee, and N. Howgrave-Graham, "Factoring $N = p^r q$ for large r", Advances in Cryptology – Crypto '99, Lecture Notes in Computer Science Vol. 1666, Springer-Verlag, pp. 326–337, 1999
5. J.-S. Coron, "Finding Small Roots of Bivariate Integer Polynomial Equations Revisited", Advances in Cryptology – Eurocrypt '04, Lecture Notes in Computer Science Vol. 3027, Springer-Verlag, pp. 492–505,2004

6. D. Coppersmith, "Finding a Small Root of a Univariate Modular Equation", Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science Vol. 1070, Springer-Verlag, pp. 155–165, 1996
7. D. Coppersmith, "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known", Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science Vol. 1070, Springer-Verlag, pp. 178–189, 1996
8. D. Coppersmith, "Small solutions to polynomial equations and low exponent vulnerabilities", Journal of Cryptology, Vol. 10(4), pp. 223–260, 1997.
9. D. Coppersmith, "Finding Small Solutions to Small Degree Polynomials", Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Volume 2146, Springer-Verlag, pp. 20–31, 2001.
10. N. Howgrave-Graham, "Finding small roots of univariate modular equations revisited", Proceedings of Cryptography and Coding, Lecture Notes in Computer Science Vol. 1355, Springer-Verlag, pp. 131–142, 1997
11. N. Howgrave-Graham, "Approximate Integer Common Divisors", Cryptography and Lattice Conference (CaLC 2001), Lecture Notes in Computer Science Vol. 2146, Springer-Verlag, pp. 51–66, 2001
12. A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," Mathematische Annalen, Vol. 261, pp. 513–534, 1982
13. A. May, "Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring", Advances in Cryptology – Crypto '04, Lecture Notes in Computer Science Vol. 3152, Springer Verlag, pp. 213–219, 2004
14. A. May, "Secret Exponent Attacks on RSA-type Schemes with Moduli $N = p^r q$", Practice and Theory in Public Key Cryptography – PKC 2004, Lecture Notes in Computer Science Vol. 2947, Springer-Verlag, pp. 218–230, 2004
15. V. Shoup, "OAEP Reconsidered", Advances in Cryptology – Crypto 2001, Lecture Notes in Computer Science Vol. 2139, Springer-Verlag, pp. 239–259, 1998

Computational Indistinguishability Between Quantum States and Its Cryptographic Application

Akinori Kawachi[1], Takeshi Koshiba[2], Harumichi Nishimura[3], and Tomoyuki Yamakami[4]

[1] Graduate School of Information Science and Engineering, Tokyo Institute of Technology, Ookayama 2-12-1, Meguro-ku, Tokyo 152-8552, Japan
kawachi@is.titech.ac.jp

[2] Secure Computing Laboratory, Fujitsu Laboratories Ltd., 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki 211-8588, Japan
koshiba@acm.org

[3] ERATO Quantum Computation and Information Project, Japan Science and Technology Agency, Matsuo-bldg.2F, 406 Iseya-cho, Kamigyo-ku, Kyoto 602-0873, Japan
hnishimura@qci.jst.go.jp

[4] Computer Science Program, Trent University, Peterborough, Ontario, Canada K9J 7B8
TomoyukiYamakami@TrentU.CA

Abstract. We introduce a problem of distinguishing between two quantum states as a new underlying problem to build a computational cryptographic scheme that is "secure" against quantum adversary. Our problem is a natural generalization of the distinguishability problem between two probability distributions, which are commonly used in computational cryptography. More precisely, our problem QSCD_{ff} is the computational distinguishability problem between two types of random coset states with a hidden permutation over the symmetric group. We show that (i) QSCD_{ff} has the trapdoor property; (ii) the average-case hardness of QSCD_{ff} coincides with its worst-case hardness; and (iii) QSCD_{ff} is at least as hard in the worst case as the graph automorphism problem. Moreover, we show that QSCD_{ff} cannot be efficiently solved by any quantum algorithm that naturally extends Shor's factorization algorithm. These cryptographic properties of QSCD_{ff} enable us to construct a public-key cryptosystem, which is likely to withstand any attack of a polynomial-time quantum adversary.

1 Introduction

Since Diffie and Hellman [15] first used a computationally intractable problem to build a key exchange protocol, computational cryptography has been extensively investigated; especially, a number of practical cryptographic systems (e.g., public-key cryptosystems (PKCs), bit commitment schemes (BCSs), pseudorandom gen-

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 268–284, 2005.

erators, and digital signature schemes) have been constructed under reasonable computational assumptions, such as the hardness of the integer factorization problem (IFP) and the discrete logarithm problem (DLP), where we have not found any efficient classical (deterministic or probabilistic) algorithm. Nevertheless, if an adversary runs a *quantum computer* (we call such an adversary a *quantum adversary*), he can efficiently solve various problems, including IFP (and quadratic residuosity problem) [40], DLP (and Diffie-Hellman problem) [10, 26, 40], and the principal ideal problem [22]. Therefore, the quantum adversary can easily break any cryptosystem whose security relies on the hardness of these problems.

A new area of cryptography, so-called *quantum cryptography*, has emerged to deal with quantum adversary and has been dramatically developed over the past two decades. In 1984, Bennett and Brassard [7] proposed a *quantum key distribution* scheme, which is a key distribution protocol using quantum communication. Later, Mayers [33] proved its unconditional security. Nevertheless, Mayers [32] and Lo and Chau [30] independently demonstrated that quantum mechanics cannot necessarily make all cryptographic schemes information-theoretically secure. In particular, they proved that no quantum BCS can be both concealing and binding unconditionally. Therefore, it is still important to take "computational" approaches to quantum cryptography. In the literature, there are a number of quantum cryptographic properties discussed from the complexity-theoretic point of view [1, 12, 13, 14, 16, 36].

Recall that a quantum computer is capable of breaking many computational assumptions on which the security of existing cryptographic protocols rely. To build a secure cryptosystem against any attack of a quantum adversary, it is important to discover computationally-hard problems that can be used as a building block of the cryptosystem. For example, the subset sum (knapsack) problem and the shortest vector problem are used as a basis of knapsack-based cryptosystems [24, 36] and lattice-based cryptosystems [4, 38]. Although quantum adversaries are currently ineffective in the attack on these cryptosystems, it is unknown whether they can essentially withstand quantum adversaries. We therefore continue searching for better underlying problems to build quantum cryptosystems which can withstand any attack of quantum adversaries. We discuss this issue in depth in Section 1.2.

This paper proposes a *new* problem, called $\mathrm{QSCD}_{\mathit{ff}}$ (quantum state computational distinguishability with fully flipped permutations), which satisfies useful cryptographic properties to build a quantum cryptosystem. Our problem $\mathrm{QSCD}_{\mathit{ff}}$ generalizes the distinguishability problems between two probability distributions used in [8, 18, 43].

Definition 1. The *advantage* of a polynomial-time quantum algorithm $\mathcal{A}$ that distinguishes between two l-qubit states ρ_0 and ρ_1 is the function $\delta(l)$ defined as:

$$\delta(l) = \left| \Pr_{\mathcal{A}}[\mathcal{A}(\rho_0) = 1] - \Pr_{\mathcal{A}}[\mathcal{A}(\rho_1) = 1] \right|,$$

where the subscript $\mathcal{A}$ means that outputs of $\mathcal{A}$ are determined randomly by measuring the final state of $\mathcal{A}$ on the computational basis. The distinguishability

problem between ρ_0 and ρ_1 is said to be *solvable by $\mathcal{A}$ with absolute (infinitely-often, resp.) advantage $\delta(l)$* if the above equation holds for any sufficiently large (infinitely many, resp.) number l.

The problem QSCD_{ff} is defined as the distinguishability problem between two random coset states ρ_π^+ and ρ_π^- with a hidden permutation π. Let S_n be the symmetric group of degree n and let $\mathcal{K}_n = \{\pi \in S_n : \pi^2 = id \text{ and } \forall i \in \{1, ..., n\}[\pi(i) \neq i]\}$, where n is described as $2(2k+1)$ for some $k \in \mathbb{N}$.

Definition 2. QSCD_{ff} is the distinguishability problem between the following two quantum states:

$$\rho_\pi^+ = \frac{1}{2n!} \sum_{\sigma \in S_n} (|\sigma\rangle + |\sigma\pi\rangle)(\langle\sigma| + \langle\sigma\pi|) \text{ and } \rho_\pi^- = \frac{1}{2n!} \sum_{\sigma \in S_n} (|\sigma\rangle - |\sigma\pi\rangle)(\langle\sigma| - \langle\sigma\pi|),$$

where $\pi \in \mathcal{K}_n$.

The parameter n of the above definition is used to measure the computational complexity of our problem and is called the *security parameter* in the cryptographic context. From a technical reason, this security parameter must be of the form $2(2k+1)$ for a certain $k \in \mathbb{N}$ as stated above. Moreover, we assume that any permutation σ can be represented in binary using $O(n \log n)$ bits.

1.1 Our Contributions

This paper shows three cryptographic properties of QSCD_{ff} and its application to quantum cryptography. These properties are summarized as follows: (1) QSCD_{ff} has the trapdoor property; namely, given a hidden permutation π, we can efficiently distinguish between ρ_π^+ and ρ_π^-; (2) the average-case hardness of QSCD_{ff} over randomly chosen permutations $\pi \in \mathcal{K}_n$ coincides with its worst-case hardness; and (3) the hardness of QSCD_{ff} is lower-bounded by the worst-case hardness of the graph automorphism problem, defined as

Graph Automorphism Problem: (GA)
input: an undirected graph $G = (V, E)$;
output: YES if G has a non-trivial automorphism, and NO otherwise.

Since GA is not known to be solved efficiently, QSCD_{ff} seems hard to solve. Moreover, we show that QSCD_{ff} cannot be efficiently solved by any quantum algorithm that naturally extends Shor's factorization algorithm.

Technically speaking, the cryptographic properties of QSCD_{ff} follows mainly from the definition of the set $\mathcal{K}_n$ of the hidden permutations. Although the definition seems somewhat artificial, the following properties of $\mathcal{K}_n$ lead to cryptographic and complexity-theoretic properties of QSCD_{ff}: (i) $\pi \in \mathcal{K}_n$ is of order 2, which provides the trapdoor property of QSCD_{ff}. (ii) For any $\pi \in \mathcal{K}_n$, the conjugacy class of π is equal to $\mathcal{K}_n$, which enables us to prove the equivalence between the worst-case/average-case hardness of QSCD_{ff}. (iii) GA is (polynomial-time Turing) equivalent to its subproblem with the promise that a given graph has a unique non-trivial automorphism in $\mathcal{K}_n$ or none at all. This equivalence is

exploited to give a complexity-theoretic lower bound of QSCD_{ff}, that is, the worst-case hardness of GA. For these proofs, we introduce new techniques: a new version of the so-called *coset sampling method*, which is broadly used in extensions of Shor's algorithm (see, e.g., [37]) and a quantum version of the hybrid argument, which is a strong tool for security reduction in modern cryptography.

As for an application of QSCD_{ff}, we also construct a public-key cryptosystem. Several advantages of using QSCD_{ff} will be discussed in depth in Section 1.2.

1.2 Comparison Between Our Work and Previous Work

In recent literature, computational-complexity aspects of quantum states have been spotlighted in connection to quantum cryptography. For instance, the notion of statistical distinguishability between quantum states was investigated by Watrous [42] and Kobayashi [27] in the context of quantum zero-knowledge proofs. They proved that certain problems of statistically distinguishing between two quantum states are promise-complete for quantum zero-knowledge proof systems. Aharonov and Ta-Shma [2] also studied the computational complexity of quantum-state generation and showed its connection to quantum adiabatic computing and statistical zero-knowledge proofs.

Our distinguishability problem QSCD_{ff} is also rooted in computational complexity theory. In this subsection, we briefly discuss various advantages of using QSCD_{ff} as a basis of quantum cryptosystems in comparison with other existing cryptosystems and their underlying problems.

Average-case Hardness versus Worst-case Hardness. In general, the efficient solvability of a problem on average does not guarantee that the problem can be solved efficiently by a worst-case algorithm. It is therefore desirable to show that the average-case hardness of cracking a cryptographic system is equivalent to its worst-case hardness. Unfortunately, there are few cryptographic problems known to be reduced from average-case hardness to worst-case hardness.

There are two types of worst-case/average-case reductions discussed in the literature. The first one is a strong reduction, which transforms an arbitrary instance of length n to a random instance of the same length or length polynomial in n. Ajtai [3] found a remarkable connection between the average-case and the worst-case hardness for certain versions of the shortest vector problem (SVP) in this strong sense. He showed an efficient reduction from the problem of approximating the shortest vector in a given n-dimensional lattice in the worst case to the approximation problem of the shortest vectors in a random lattice over a certain class of lattices with a larger polynomial approximation factor in n. A reduction between average-case and worst-case hardness has since then been extensively studied. Micciancio and Regev [34], for instance, gave the average-case/worst-case connection factor of approximately n for approximating SVP (see [9] by Bogdanov and Trevisan and references therein for general results with respect to worst-case/average-case reductions).

The second type of reduction is a weak reduction of Tompa and Woll [41], where the reduction is randomized only over part of its instances. A typical example is DLP, which can be randomly reduced to itself by a reduction that

maps instances to not all instances of the same length but rather all instances of the same underlying group. It is, nonetheless, unknown that there exists a reduction from DLP with the worst-case prime to DLP with a random prime.

In this paper, we show that QSCD_{ff} has a worst-case/average-case reduction of the first kind. Our reduction depends only on the length of the instance unlike a reduction for DLP and the average-case hardness of QSCD_{ff} coincides with its worst-case hardness unlike reductions for lattice problems. Note that DLP and the inverting problem of the RSA function, whose worst-case/average-case reductions are of the second kind, can be efficiently solved in the worst case by Shor's algorithm [40]. The graph isomorphism problem (GI) and GA—well-known graph-theoretical problems—also have the connection of the second kind [41]. Although no efficient quantum algorithm is discovered yet for them, there is no known cryptographic system whose security are reduced from them. Our distinguishability problem QSCD_{ff} is the first *cryptographic* problem with the worst-case/average-case reduction of the first kind, which has not been solved efficiently on a quantum computer.

Most problems seem to lack any strong connection between their average-case harness and worst-case hardness. In particular, there is no known cryptographic system that is based on the worst-case hardness of the subset sum problem or its subproblems.

Exponential time versus Subexponential time. The *hidden subgroup problem* (HSP) has been a central issue discussed for both positive and negative aspects of the power of quantum computation. Both IFP and DLP can be viewed as special cases of HSP on Abelian groups (AHSP). Kitaev [26] showed that AHSP can be efficiently solved. He introduced a polynomial-time algorithm for the quantum Fourier transform on Abelian groups, which is a generalization of the original quantum Fourier transform used in Shor's algorithm [40]. Although AHSP can be efficiently solved, the more general non-Abelian group case is unlikely to be solved by simply applying currently known techniques. (Some special non-Abelian group cases were studied in [17, 20, 23, 29, 35, 37].) Another important variant is the HSP on the dihedral groups (DHSP). Recently, Regev [37] demonstrated a quantum reduction from the unique shortest vector problem (uSVP) to a slightly different variant of DHSP. Note that uSVP is used in lattice-based PKCs [4, 38]. Moreover, Kuperburg [29] gave a subexponential-time quantum algorithm for DHSP. Although these results do not directly imply a subexponential-time quantum algorithm for uSVP, they may be an important clue to find the desired algorithm.

Our problem QSCD_{ff} is closely related to a much harder problem: HSP on the symmetric groups (SHSP). No subexponential-time quantum algorithm is known for SHSP. A distinguishability problem, similar to QSCD_{ff}, defined in terms of SHSP was introduced by Hallgren, Russell and Ta-Shma [23], who showed that any standard algorithm[1] takes exponential time to solve their problem. Here,

[1] The algorithms that run an essential part of Shor's algorithm [40] are simply called *standard methods*.

we show that their problem is polynomial-time reducible to QSCD_{ff}. This immediately implies that that any standard algorithm that solves QSCD_{ff} also requires exponential time. The hardness result of Hallgren et al. was recently strengthened by Grigni et al. [20] and Kempe and Shalev [25]. Finding even a subexponential algorithm for QSCD_{ff} seems a daunting task. On the contrary, this suggests that our problem QSCD_{ff} is more reliable than, e.g., uSVP. This situation is similar to the case of DLP over different groups on classical computation. DLP over $\mathbb{Z}_p^*$ (p is a prime) is classically solved in subexponential time whereas there is no known classical subexponential-time algorithm for DLP over certain groups used in elliptic curve cryptography. It is believed that DLP over such groups is more reliable than DLP over $\mathbb{Z}_p^*$.

We prove that the computational complexity of QSCD_{ff} is lower-bounded by that of GA, which is not known to be in $\mathrm{NP} \cap \text{co-NP}$. Well-known upper bounds of GA are $\mathrm{NP} \cap \text{co-AM}$ [19, 39], SPP [5], and UAP [11]. To our best knowledge, most cryptographic problems fall in $\mathrm{NP} \cap \text{co-NP}$ and few cryptographic systems are lower-bounded by the worst-case hardness of the problems not known to be in $\mathrm{NP} \cap \text{co-NP}$.

Quantum Computational Cryptography. Quantum key distribution gives a foundation to symmetric-key cryptosystems (SKCs). For instance, the quantum key distribution scheme in [7] achieves unconditionally secure sharing of secret keys for SKCs using an authenticated classical communication channel. Both SKCs and PKCs have their own advantages and disadvantages. For instance, PKCs save a number of secret keys compared with SKCs in a large network; however, they need computational assumptions for their security and is vulnerable to, for instance, the man-in-the-middle attack. As an application of QSCD_{ff}, we propose a new computational quantum PKC whose security relies on the computational hardness of QSCD_{ff}.

Of many existing PKCs, few make their security solely based on the worst-case hardness of their underlying problems. Quantum adversaries can break many PKCs whose underlying problems are number-theoretic problems because these problems are solvable by efficient quantum algorithms. Recently, Okamoto, Tanaka, and Uchiyama [36] proposed a quantum analogue of PKCs based on a certain subset of the knapsack problem and showed that their cryptosystem withstands certain known attacks of a quantum adversary. Our quantum PKC also seems to resist a quantum adversary since we can prove the existence of a security reduction from the problem GA, which is not known to be solved efficiently even on a quantum computer.

2 Cryptographic Properties of QSCD_{ff}

We show three cryptographic properties of QSCD_{ff} introduced in the previous section. These properties will help us construct a cryptographic system in Section 3. Hereafter, we assume the reader's familiarity with basics of quantum computation. Recall the two quantum states $\rho_\pi^+ = \frac{1}{2n!}\sum_{\sigma\in S_n}(|\sigma\rangle + |\sigma\pi\rangle)(\langle\sigma| + \langle\sigma\pi|)$ and $\rho_\pi^- = \frac{1}{2n!}\sum_{\sigma\in S_n}(|\sigma\rangle - |\sigma\pi\rangle)(\langle\sigma| - \langle\sigma\pi|)$ for a hidden permutation $\pi \in \mathcal{K}_n$.

For simplicity, let ι denote the maximally mixed state, i.e., $\iota = \frac{1}{n!}\sum_{\sigma\in S_n}|\sigma\rangle\langle\sigma|$, which appears later as a technical tool.

2.1 Trapdoor Property

We prove that QSCD_{ff} has the *trapdoor property*, which plays a key role in various cryptosystems. We present an efficient distinction algorithm between ρ_π^+ and ρ_π^- with a hidden permutation π in $\mathcal{K}_n$.

Theorem 1. There exists a polynomial-time quantum algorithm that, given $\pi \in \mathcal{K}_n$, distinguishes between ρ_π^+ and ρ_π^- with certainty.

Proof. Let χ be any given unknown state, which is either ρ_π^+ or ρ_π^-. The desired distinction algorithm for χ is given as follows.

(D1) Prepare two quantum registers: the first register holds a control bit and the second one holds χ. Apply the Hadamard transformation H to the first register. The state of the system now becomes $H|0\rangle\langle 0|H \otimes \chi$.

(D2) Apply the Controlled-π operator C_π to the two registers, where $C_\pi|0\rangle|\sigma\rangle = |0\rangle|\sigma\rangle$ and $C_\pi|1\rangle|\sigma\rangle = |1\rangle|\sigma\pi\rangle$ for any $\sigma \in S_n$. Since $\pi^2 = id$ for any $\pi \in \mathcal{K}_n$, the state of the entire system is expressed as $\frac{1}{n!}\sum_{\sigma\in S_n}|\psi^+_{\pi,\sigma}\rangle\langle\psi^+_{\pi,\sigma}|$ if $\chi = \rho_\pi^+$ and $\frac{1}{n!}\sum_{\sigma\in S_n}|\psi^-_{\pi,\sigma}\rangle\langle\psi^-_{\pi,\sigma}|$ if $\chi = \rho_\pi^-$, where

$$|\psi^\pm_{\pi,\sigma}\rangle = C_\pi\left(\frac{1}{2}|0\rangle\left(|\sigma\rangle \pm |\sigma\pi\rangle\right) + |1\rangle\left(|\sigma\rangle \pm |\sigma\pi\rangle\right)\right)$$
$$= \frac{1}{2}|0\rangle(|\sigma\rangle \pm |\sigma\pi\rangle) + \frac{1}{2}|1\rangle(|\sigma\pi\rangle \pm |\sigma\rangle).$$

(D3) Apply the Hadamard transformation to the first register. If χ is ρ_π^+ and ρ_π^-, then the state of the system becomes $(H\otimes I)|\psi^+_{\pi,\sigma}\rangle = \frac{1}{\sqrt{2}}|0\rangle\left(|\sigma\rangle + |\sigma\pi\rangle\right)$ and $(H\otimes I)|\psi^-_{\pi,\sigma}\rangle = \frac{1}{\sqrt{2}}|1\rangle\left(|\sigma\rangle - |\sigma\pi\rangle\right)$, respectively. Measure the first register on the computational basis. If the result is 0, output YES; otherwise, output NO. Clearly, we obtain the correct answer with probability 1. □

2.2 Reduction from the Worst Case to the Average Case

We reduce the worst-case hardness of QSCD_{ff} to its average-case hardness. Such a reduction implies that QSCD_{ff} with a random π is at least as hard as QSCD_{ff} with the most difficult π.

Theorem 2. Assume that there exists a polynomial-time quantum algorithm $\mathcal{A}$ that solves QSCD_{ff} with absolute (infinitely-often, resp.) non-negligible advantage for a uniformly random $\pi \in \mathcal{K}_n$; namely, there exists a polynomial p such that, for any sufficiently large (infinitely many, resp.) number n,

$$\left|\Pr_{\pi,\mathcal{A}}[\mathcal{A}(\rho_\pi^+) = 1] - \Pr_{\pi,\mathcal{A}}[\mathcal{A}(\rho_\pi^-) = 1]\right| > 1/p(n),$$

where π is chosen uniformly at random from $\mathcal{K}_n$. Then, there exists a polynomial-time quantum algorithm $\mathcal{B}$ that solves QSCD_{ff} with absolute (infinitely-often, resp.) non-negligible advantage in the worst case.

Proof. Let χ be any given state, which is either ρ_π^+ or ρ_π^-. The desired worst-case algorithm $\mathcal{B}$ is built from the average-case algorithm $\mathcal{A}$ in the following way.

(R1) Choose a permutation $\tau \in S_n$ uniformly at random and then multiply χ by τ from the right. If $\chi = \rho_\pi^+$, then we obtain the quantum state

$$\chi' = \frac{1}{2n!} \sum_{\sigma \in S_n} (|\sigma\tau\rangle + |\sigma\tau\tau^{-1}\pi\tau\rangle)(\langle\sigma\tau| + \langle\sigma\tau\tau^{-1}\pi\tau|)$$
$$= \frac{1}{2n!} \sum_{\sigma' \in S_n} (|\sigma'\rangle + |\sigma'\tau^{-1}\pi\tau\rangle)(\langle\sigma'| + \langle\sigma'\tau^{-1}\pi\tau|).$$

If $\chi = \rho_\pi^-$, then we obtain $\chi' = \dfrac{1}{2n!} \displaystyle\sum_{\sigma \in S_n} (|\sigma\rangle - |\sigma\tau^{-1}\pi\tau\rangle)(\langle\sigma| - \langle\sigma\tau^{-1}\pi\tau|)$.

(R2) Invoke the average-case algorithm $\mathcal{A}$ on the input χ'.

(R3) Output the outcome of $\mathcal{A}$.

Note that $\tau^{-1}\pi\tau \in \mathcal{K}_n$ for any τ and there exists a $\tau \in S_n$ satisfying that $\tau^{-1}\pi\tau = \pi'$ for any $\pi' \in \mathcal{K}_n$. Hence, the conjugacy class of π is equal to $\mathcal{K}_n$. Moreover, the number of all $\tau \in S_n$ for which $\tau^{-1}\pi\tau = \pi'$ is independent of the choice of $\pi' \in \mathcal{K}_n$. From these properties, $\tau^{-1}\pi\tau$ is uniformly distributed over $\mathcal{K}_n$. Therefore, feeding the input χ' to algorithm $\mathcal{A}$ guarantees the non-negligible advantage. □

2.3 Hardness of QSCD_{ff}

We show that the computational complexity of QSCD_{ff} is lower-bounded by that of GA by constructing an efficient reduction from GA to QSCD_{ff}. Our reduction constitutes two parts: a reduction from GA to a variant of GA, called $\mathrm{UniqueGA}_{ff}$, and a reduction from $\mathrm{UniqueGA}_{ff}$ to QSCD_{ff}. We also discuss a relationship between QSCD_{ff} and SHSP, which suggests that QSCD_{ff} may be hard for polynomial-time quantum algorithms to solve.

To describe the desired reduction, we begin with introducing two variants of GA. Earlier, Köbler, Schöning and Torán [28] introduced the following *unique graph automorphism problem* (UniqueGA).

Unique Graph Automorphism Problem: (UniqueGA)
input: an undirected graph $G = (V, E)$;
promise: G has a unique non-trivial automorphism or no non-trivial automorphisms;
output: YES if G has the non-trivial automorphism, and NO otherwise.

Notice that UniqueGA is called (1GA, GA) as a promise problem in [28]. In connection to our distinguishability problem, we introduce the *unique graph automorphism with fully-flipped permutation* ($\mathrm{UniqueGA}_{ff}$), which plays an important role in the reduction.

Unique Graph Automorphism with Fully-Flipped Permutation: (UniqueGA$_{ff}$)
input: an undirected graph $G = (V, E)$, where $|V| = n = 2(2k + 1)$ for some $k \in \mathbb{N}$;
promise: G has a unique non-trivial automorphism $\pi \in \mathcal{K}_n$, or no non-trivial automorphisms;
output: YES if G has the non-trivial automorphism, and NO otherwise.

Next, we discuss the so-called *coset sampling method*, which has been largely used in many extensions of Shor's algorithm.

Lemma 1. There exists a polynomial-time quantum algorithm that, given an instance G of UniqueGA$_{ff}$, generates a quantum state ρ_π^+ if G is an "YES" instance with its unique non-trivial automorphism π, or $\iota = \frac{1}{n!}\sum_{\sigma\in S_n} |\sigma\rangle\langle\sigma|$ if G is a "NO" instance.

Proof. Given an instance G of UniqueGA$_{ff}$, we first prepare the quantum state $\frac{1}{\sqrt{n!}}\sum_{\sigma\in S_n} |\sigma\rangle|\sigma(G)\rangle$, where $\sigma(G)$ is the graph resulting from by relabeling its nodes according to a permutation σ. By discarding the second register, we obtain the unique quantum state χ in the first register. Then, $\chi = \rho_\pi^+$ if G is an "YES" instance with the unique non-trivial automorphism π, and $\chi = \iota$ otherwise, as requested. □

Now, we introduce a new version of the coset sampling method as a technical tool for our reduction. Note that this algorithm essentially requires the fact that the hidden π is an odd permutation, which is one of the special properties of $\mathcal{K}_n$.

Lemma 2. There exists a polynomial-time quantum algorithm that, given an instance G of UniqueGA$_{ff}$, generates a quantum state ρ_π^- if G is an "YES" instance with the unique non-trivial automorphism π, or ι if G is a "NO" instance.

Proof. Similar to the algorithm of Lemma 1, we prepare the quantum state $\frac{1}{\sqrt{n!}}\sum_{\sigma\in S_n} |\sigma\rangle|\sigma(G)\rangle$. Next, we compute the sign of each permutation in the first register and then invert its phase if the permutation is odd. We obtain the quantum state $\frac{1}{\sqrt{n!}}\sum_{\sigma\in S_n}(-1)^{\mathrm{sgn}(\sigma)}|\sigma\rangle|\sigma(G)\rangle$, where $\mathrm{sgn}(\sigma) = 0$ if σ is even, and $\mathrm{sgn}(\sigma) = 1$ otherwise. By discarding the second register, we can obtain a quantum state χ in the first register. Note that, since π is odd, if σ is odd (even, resp.) then $\sigma\pi$ is even (odd, resp.). Therefore, $\chi = \rho_\pi^-$ if G is an "YES" instance with the unique non-trivial automorphism π, and $\chi = \iota$ otherwise. □

We are now ready to present a reduction from GA to QSCD$_{ff}$, which implies that QSCD$_{ff}$ is computationally at least as hard as GA.

Theorem 3. If there exists a polynomial-time quantum algorithm that solves QSCD$_{ff}$ with absolute non-negligible advantage, there exists a polynomial-time quantum algorithm that solves any instance of GA in the worst case with non-negligible probability.

Proof. We first show that GA is polynomial-time Turing equivalent to UniqueGA$_{ff}$ and then give a reduction from UniqueGA$_{ff}$ to QSCD$_{ff}$. The reduction from GA to UniqueGA$_{ff}$ is similar to the one given by Köbler, Schöning and Torán [28], who presented a polynomial-time Turing reduction from GA to UniqueGA. Their polynomial-time algorithm for GA invokes UniqueGA as an oracle with a promised input, that is, a graph with even number of nodes which has either the unique non-trivial automorphism without fixed points or no non-trivial automorphisms. Carefully reading the construction of their reduction, we can easily modify it to fit our reduction from GA to UniqueGA$_{ff}$. Moreover, slightly modifying the gadgets for their reduction, we can satisfy the condition that $n = 2(2k+1)$ for some $k \in \mathbb{N}$. Thus, we obtain the following lemma.

Lemma 3. UniqueGA$_{ff}$ is polynomial-time Turing equivalent[2] to GA.

The complete proof of this lemma is placed in Appendix. It therefore suffices to show a reduction from UniqueGA$_{ff}$ to QSCD$_{ff}$. Assume that there exists a polynomial-time quantum algorithm $\mathcal{A}$ that solves QSCD$_{ff}$ with absolute non-negligible advantage. For a given instance G of UniqueGA$_{ff}$, we perform the following procedure:

(S1) Generate two sequences $S^+ = (\chi^+, ..., \chi^+)$ and $S^- = (\chi^-, ..., \chi^-)$ of $8p^2(n)n$ quantum states from G using the algorithms of Lemmas 1 and 2, respectively.
(S2) Invoke $\mathcal{A}$ on each component in S^+ and S^- as an input. Let $R^+ = (\mathcal{A}(\chi^+), ..., \mathcal{A}(\chi^+))$ and $R^- = (\mathcal{A}(\chi^-), ..., \mathcal{A}(\chi^-))$ be the resulting sequences.
(S3) Output YES if the gap between the numbers of 1's in R^+ and R^- is at least $4p(n)n$, output NO otherwise.

Note that if G is an "YES" instance, $S^+ = (\overbrace{\rho_\pi^+, ..., \rho_\pi^+}^{8p^2(n)n})$ and $S^- = (\overbrace{\rho_\pi^-, ..., \rho_\pi^-}^{8p^2(n)n})$, otherwise $S^+ = S^- = \overbrace{(\iota, ..., \iota)}^{8p^2(n)n}$. Therefore, if G is an "YES" instance, then there is a gap between the numbers of 1's in R^+ and in R^- because of the property of $\mathcal{A}$; otherwise, there is no gaps between them.

We now estimate this gap by the Hoeffding bound. Let X^+ and X^- be two random variables expressing the numbers of 1's in R^+ and in R^-, respectively. If G is an "YES" instance, $\Pr[|X^+ - X^-| > 4p(n)n] > 1 - 2e^{-n}$ by the Hoeffding bound since $|\Pr[\mathcal{A}(\rho_\pi^+) = 1] - \Pr[\mathcal{A}(\rho_\pi^-) = 1]| > 1/p(n)$. Similarly, if G is a "NO" instance, $\Pr[|X^+ - X^-| < 4p(n)n] > 1 - 2e^{-n}$. This guarantees the above procedure to solve UniqueGA$_{ff}$ efficiently, as requested. □

As stated in Section 1, the distinguishability problem QSCD$_{ff}$ is rooted in SHSP. It is shown that a natural extension of Shor's algorithm cannot solve

[2] If a Turing reduction to a promise problem makes only queries that satisfy the promise, the reduction is called *smart* [21]. Smart reductions are desirable for security reductions. The reduction from GA to UniqueGA in [28] is indeed smart and thus, so is this reduction.

the distinguishability problem between ρ_π^+ and ι in [23, 20, 25]. Here, we give a theorem on a relationship between QSCD_{ff} and the distinguishability problem between ρ_π^+ and ι.

Before stating the theorem, we give a conversion algorithm for ρ_π^+ and ρ_π^-. This algorithm will be used in the proof of the theorem as well as the construction of a PKC in the subsequent section.

Lemma 4. There exists a polynomial-time quantum algorithm that converts ρ_π^+ into ρ_π^- and keeps ι as it is with certainty.

Proof. Given ρ_π^+, the desired algorithm inverts its phase according to the sign of the permutation by performing the following transformation:

$$|\sigma\rangle + |\sigma\pi\rangle \longmapsto (-1)^{\mathrm{sgn}(\sigma)}|\sigma\rangle + (-1)^{\mathrm{sgn}(\sigma\pi)}|\sigma\pi\rangle.$$

Recall that $\mathrm{sgn}(\sigma) = 0$ if σ is even and $\mathrm{sgn}(\sigma) = 1$ otherwise. Note that deciding the sign of a given permutation takes only polynomial time. Since π is odd, the above algorithm converts ρ_π^+ into ρ_π^-. Clearly, the algorithm does not alter the quantum state ι. □

The following theorem implies that QSCD_{ff} cannot be efficiently solved by any algorithm that naturally extends Shor's factoring algorithm. To prove the theorem, we need a quantum version of the so-called *hybrid argument*.

Theorem 4. If there exists a polynomial-time quantum algorithm that solves QSCD_{ff} with absolute (infinitely-often, resp.) non-negligible advantage, then there exists a polynomial-time quantum algorithm that solves the distinguishability problem between ρ_π^+ and ι with absolute (infinitely-often, resp.) non-negligible advantage.

Proof. We prove only the absolute advantage case. Assume that a polynomial-time quantum algorithm $\mathcal{A}$ solves QSCD_{ff} with absolute non-negligible advantage; namely, there exist a number $n_0 \geq 1$ and a polynomial $q(n)$ such that

$$\left|\Pr_{\mathcal{A}}[\mathcal{A}(\rho_\pi^+) = 1] - \Pr_{\mathcal{A}}[\mathcal{A}(\rho_\pi^-) = 1]\right| > 1/q(n)$$

for all numbers $n \geq n_0$. Let $\mathcal{A}'$ be the algorithm that applies the conversion algorithm of Lemma 4 to a given state χ ($= \rho_\pi^+$ or ι) and then feeds the resulting state χ' ($= \rho_\pi^-$ or ι) to $\mathcal{A}$. Note that $\mathcal{A}'(\rho_\pi^+) = \mathcal{A}(\rho_\pi^-)$ and $\mathcal{A}'(\iota) = \mathcal{A}(\iota)$. It immediately follows by the triangle inequality that, for any number $n \geq n_0$,

$$\left|\Pr_{\mathcal{A}}[\mathcal{A}(\rho_\pi^+) = 1] - \Pr_{\mathcal{A}}[\mathcal{A}(\iota) = 1]\right| + \left|\Pr_{\mathcal{A}'}[\mathcal{A}'(\rho_\pi^+) = 1] - \Pr_{\mathcal{A}'}[\mathcal{A}'(\iota) = 1]\right| > 1/q(n).$$

This inequality implies that, for each number $n \geq n_0$, we obtain either

$$\left|\Pr_{\mathcal{A}}[\mathcal{A}(\rho_\pi^+) = 1] - \Pr_{\mathcal{A}}[\mathcal{A}(\iota) = 1]\right| > 1/2q(n)$$

or

$$\left|\Pr_{\mathcal{A}'}[\mathcal{A}'(\rho_\pi^+) = 1] - \Pr_{\mathcal{A}'}[\mathcal{A}'(\iota) = 1]\right| > 1/2q(n).$$

The desired algorithm $\mathcal{B}$ first chooses either $\mathcal{A}$ or $\mathcal{A}'$ at random and then simulates the chosen algorithm. Obviously, this algorithm solves the distinguishability problem between ρ_π^+ and ι with absolute non-negligible advantage, completing the proof. □

3 Application

We have shown useful cryptographic properties of QSCD_{ff}. As an application of QSCD_{ff}, we build a quantum public-key cryptosystem (PKC) whose security relies on the hardness of QSCD_{ff}. First, we give an efficient quantum algorithm that generates ρ_π^+ from π.

Lemma 5. There exists a polynomial-time quantum algorithm that, given $\pi \in \mathcal{K}_n$, generates the quantum state ρ_π^+ with certainty.

Proof. The desired generation algorithm uses two registers and is given as follows. The correctness of the algorithm is obvious.

(G1) Choose a permutation σ from S_n uniformly at random and store it in the second register. Then, the entire system is in the state $|0\rangle|\sigma\rangle$.
(G2) Apply the Hadamard transformation to the first register: $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\sigma\rangle$.
(G3) Apply the Controlled-π to the both registers: $\frac{1}{\sqrt{2}}(|0\rangle|\sigma\rangle + |1\rangle|\sigma\pi\rangle)$.
(G4) Apply the Hadamard transformation to the first register again: $\frac{1}{2}((|0\rangle + |1\rangle)|\sigma\rangle + (|0\rangle - |1\rangle)|\sigma\pi\rangle)$.
(G5) Measure the first register on the computational basis. If 0 is observed, then the quantum state in the second register is ρ_π^+. Otherwise, the state of the second register is ρ_π^-. Now, apply the conversion algorithm given in Lemma 4 to ρ_π^-. □

Next, we describe our quantum PKC and give its security proof. For the security proof, we need to specify the model of attacks. Of all attack models in [6], we pay our attention to a quantum analogue of *the indistinguishability against the chosen plaintext attack (IND-CPA)*. In particular, we adopt the weakest scenario in quantum counterparts of IND-CPA as follows.

Alice (sender) wants to send securely a classical message to Bob (receiver) via a quantum channel. Assume that Alice and Bob are polynomial-time quantum Turing machines. Bob first generates certain quantum states for encryption keys. Alice then requests Bob for his encryption keys. Note that anyone can request him for the encryption keys. Now, we assume that Eve (adversary) can pick up the encrypted messages from the quantum channel, and tries to extract the original message using her quantum computer, i.e., a polynomial-time quantum Turing machine. Since Eve can also obtain Bob's encryption keys as well as Alice does, she can exploit polynomially many encryption keys to distinguish the encrypted message. Thus, we assume that Eve attacks the protocol during the message transmission phase to reveal the content of the encrypted message.

The protocol to transmit a message using our PKC consists of two phases: the key transmission phase and the message transmission phase. We will give a reduction from the worst-case hardness of GA to such Eve's attack.

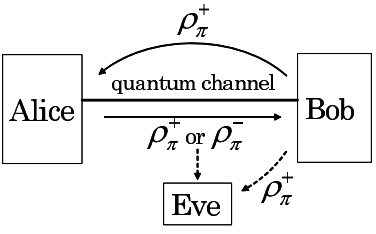

Fig. 1. Our public-key cryptosystem

We first describe the protocol of our quantum PKC as follows.
[Key transmission phase]
(A1) Bob chooses a decryption key π uniformly at random from $\mathcal{K}_n$.
(A2) Bob generates sufficiently many copies of the encryption key ρ_π^+.
(A3) Alice obtains encryption keys from Bob.
[Message transmission phase]
(A4) Alice encrypts 0 or 1 into ρ_π^+ or ρ_π^-, respectively, and sends it to Bob.
(A5) Bob decrypts Alice's message using the decryption key π.

Step (A1) can be easily implemented by uniformly choosing transpositions one by one in such a way that all transpositions are different and by forming the product of these transpositions. Step (A2) is done by the generation algorithm of Lemma 5. For Step (A4), we exploit the conversion algorithm of Lemma 4. Note that Alice sends Bob either the received state ρ_π^+ or its converted state ρ_π^- depending on Alice's bit. Finally, the distinction algorithm in Theorem 1 achieves Step (A5).

The security of our PKC is shown by reducing GA to Eve's attack during the message transmission phase. Our reduction is a modification of the reduction given in Theorem 3.

Proposition 1. Assume that there exists a polynomial-time quantum adversary $\mathcal{A}$ in the message transmission phase that, for any sufficiently large n, satisfies the following inequality

$$\left| \Pr_{\pi,\mathcal{A}}[\mathcal{A}(\rho_\pi^+, \rho_\pi^{+\otimes l(n)}) = 1] - \Pr_{\pi,\mathcal{A}}[\mathcal{A}(\rho_\pi^-, \rho_\pi^{+\otimes l(n)}) = 1] \right| > 1/p(n)$$

for a certain polynomial $l(n)$ indicating the number of the encryption keys in use by $\mathcal{A}$ and another polynomial $p(n)$. Then, there exists a polynomial-time quantum algorithm that solves any instance of GA in the worst case with non-negligible probability.

Proof. The proof immediately follows by replacing ρ_π^+, ρ_π^-, and ι in the proof of Theorem 3 with $(\rho_\pi^+, \rho_\pi^{+\otimes l(n)})$, $(\rho_\pi^-, \rho_\pi^{+\otimes l(n)})$, and $(\iota, \iota^{\otimes l(n)})$, respectively. □

4 Concluding Remarks

The computational distinguishability problem QSCD_{ff} has shown useful properties to build a computational PKC whose security is based on the computational hardness of GA. Although GA is reducible to QSCD_{ff}, the gap between the hardness of GA and that of QSCD_{ff} seems large because a combinatorial structure of its underlying graphs which GA enjoys is completely lost in QSCD_{ff}. It is therefore important to discover a classical problem, such as the problems of finding a centralizer or finding a normalizer [31], which captures the true hardness of QSCD_{ff}. Discovering an efficient quantum algorithm for QSCD_{ff} is likely to require a new tool and a new technique, which also bring a breakthrough in quantum computation. It is important to discover useful quantum states whose computational distinguishability is used for constructing a more secure cryptosystem.

Acknowledgments. The authors are grateful to Hirotada Kobayashi and Claude Crépeau for fruitful discussions, to John Watrous for useful comments on key ideas, to Donald Beaver, Louis Salvail, and the anonymous reviewers for their valuable suggestions.

References

1. M. Adcock and R. Cleve. A quantum Goldreich-Levin theorem with cryptographic applications. In *Proc. 19th Symp. Theoretical Aspects of Computer Science*, LNCS 2285, pp.323–334 (2002).
2. D. Aharonov and A. Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proc. 35th ACM Symp. Theory of Computing*, pp.20–29 (2003).
3. M. Ajtai. Generating hard instances of lattice problems. In *Proc. 28th ACM Symp. Theory of Computing*, pp.99–108 (1996).
4. M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. 29th ACM Symp. Theory of Computing*, pp.284–293 (1997).
5. V. Arvind and P. P. Kurur. Graph isomorphism is in SPP. In *Proc. 43rd IEEE Symp. Foundations of Computer Science*, pp.743–750 (2002).
6. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology – CRYPTO'98*, LNCS 1462, pp.26–45 (1998).
7. C. H. Bennett and G. Brassard. Quantum cryptography: public key distribution and coin tossing. In *Proc. IEEE International Conf. Computers, Systems, and Signal Processing*, pp.175–179 (1984).
8. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
9. A. Bogdanov and L. Trevisan. On worst-case to average-case reductions for NP problems. In *Proc. 44th IEEE Symp. Foundations of Computer Science*, pp.308–317 (2004).
10. D. Boneh and R. J. Lipton. Quantum cryptanalysis of hidden linear functions. In *Advances in Cryptology – CRYPTO'95*, LNCS 963, pp.424–437 (1995).
11. M. Crâsmaru, C. Glaßer, K. W. Regan, and S. Sengupta. A protocol for serializing unique strategies. In *Proc. 29th Symp. Mathematical Foundations of Computer Science*, LNCS 3153, pp.660–672 (2004).

12. C. Crépeau, P. Dumais, D. Mayers, and L. Salvail. Computational collapse of quantum state with application to oblivious transfer. In *Proc. 1st Theory of Cryptography Conf.*, LNCS 2951, pp.374–393 (2004).
13. C. Crépeau, F. Légaré, and L. Salvail. How to convert the flavor of a quantum bit commitment. In *Advances in Cryptology – EUROCRYPT'01*, LNCS 2045, pp.60–77 (2001).
14. I. Damgård, S. Fehr, and L. Salvail. Zero-knowledge proofs and string commitments withstanding quantum attacks. In *Advances in Cryptology – CRYPTO'04*, LNCS 3152, pp.254–272 (2004).
15. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.
16. P. Dumais, D. Mayers, and L. Salvail. Perfectly concealing quantum bit commitment from any quantum one-way permutation. In *Advances in Cryptology – EUROCRYPT 2000*, LNCS 1807, pp.300–315 (2000).
17. M. Ettinger and P. Høyer. On quantum algorithms for noncommutative hidden subgroups. *Advances in Applied Mathematics*, 25:239–251, 2000.
18. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. System Sci.*, 28(2):270–299, 1984.
19. S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof system. In *Advances in Computing Research, Vol. 5: Randomness and Computation*, pp.73–90. JAI Press, 1989.
20. M. Grigni, L. J. Schulman, M. Vazirani, and U. Vazirani. Quantum mechanical algorithms for the nonabelian hidden subgroup problem. In *Proc. 33rd ACM Symp. Theory of Computing*, pp.68–74 (2001).
21. J. Grollmann and A. L. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17(2):309–335, 1988.
22. S. Hallgren. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. In *Proc. 34th ACM Symp. Theory of Computing*, pp.653–658 (2002).
23. S. Hallgren, A. Russell, and A. Ta-Shma. The hidden subgroup problem and quantum computation using group representations. *SIAM J. Comput.*, 32(4):916–934, 2003.
24. R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.
25. J. Kempe and A. Shalev. The hidden subgroup problem and permutation group theory. In *Proc. 16th ACM-SIAM Symp. Discrete Algorithms*, 2005.
26. A. Kitaev. Quantum measurements and the Abelian stabilizer problem. quant-ph/9511026, 1995.
27. H. Kobayashi. Non-interactive quantum perfect and statistical zero-knowledge. In *Proc. 14th International Conf. Algorithms and Computation*, LNCS 2906, pp.178–188 (2003).
28. J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser Boston Inc., 1993.
29. G. Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. quant-ph/0302112, 2003.
30. H.-K. Lo and H. F. Chau. Is quantum bit commitment really possible? *Physical Review Letters*, 78(17):3410–3413, 1997.
31. E. M. Luks. Permutation groups and polynomial-time computation. *Groups and Computation*, 11:139–175, 1993.
32. D. Mayers. Unconditionally secure quantum bit commitment is impossible. *Physical Review Letters*, 78(17):3414–3417, 1997.

33. D. Mayers. Unconditional security in quantum cryptography. *J. ACM*, 48(3):351–406, 2001.
34. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measure. In *Proc. 45th IEEE Symp. Foundations of Computer Science*, pp.372–381 (2004).
35. C. Moore, D. Rockmore, A. Russell, and L. J. Schulman. The hidden subgroup problem in affine groups: basis selection in Fourier sampling. In *Proc. 15th ACM-SIAM Symp. Discrete Algorithms*, pp.1106–1115 (2004).
36. T. Okamoto, K. Tanaka, and S. Uchiyama. Quantum public-key cryptosystems. In *Advances in Cryptology – CRYPTO 2000*, LNCS 1880, pp.147–165 (2000).
37. O. Regev. Quantum computation and lattice problems. In *Proc. 43rd IEEE Symp. Foundations of Computer Science*, pp.520–529 (2002).
38. O. Regev. New lattice-based cryptographic constructions. In *Proc. 35th ACM Symp. Theory of Computing*, pp.407–416 (2003).
39. U. Schöning. Graph isomorphism is in the low hierarchy. *J. Comput. System Sci.*, 37:312–323, 1988.
40. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26:1484–1509, 1997.
41. M. Tompa and H. Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. In *Proc. 28th IEEE Symp. Foundations of Computer Science*, pp.472–482 (1987).
42. J. Watrous. Limits on the power of quantum statistical zero-knowledge. In *Proc. 43rd IEEE Symp. Foundations of Computer Science*, pp.459–468 (2002).
43. A. C.-C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE Symp. Foundations of Computer Science*, pp.80–91 (1982).

Appendix: Reduction from GA to UniqueGA$_{ff}$

In this Appendix, we prove Lemma 3. Köbler, Schöning and Torán [28] proved the polynomial-time Turing equivalence between GA and UniqueGA. We first review their reduction and then show how to modify it to obtain the reduction from GA to UniqueGA$_{ff}$. Note that the reduction from UniqueGA$_{ff}$ to GA is obvious.

We begin with a technical tool and notations. The reduction of Köbler et al. uses a technical tool called a *label* to distinguish each node of a given graph G from the others. The label j attached to node i consists of two chains, one of which is of length $2n + 3$ connected to node i and the other is of length j connected to the $n + 2$-nd node of the first chain (Fig. 2).

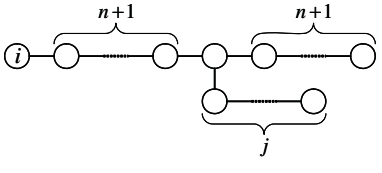

Fig. 2. Label

Note that the total size of the label j is $2n + j + 3$. Let $G_{[i]}$ denote the graph obtained from G by attaching label 1 to node i. Similarly, $G_{[i_1,\ldots,i_j]}$ is defined as

the graph with labels $1, ..., j$ respectively attached to nodes $i_1, ..., i_j$. Note that any automorphism of $G_{[i]}$ maps the node i into itself and any label adds no new automorphism into the modified graph. Let $Aut(G)$ be the automorphism group of the graph G and let $Aut(G)_{[1,...,i]}$ be the point-wise stabilizer of $\{1, ..., i\}$ in $Aut(G)$, i.e., $Aut(G)_{[1,...,i]} = \{\sigma \in Aut(G) : \forall j \in \{1, ..., i\}[\sigma(j) = j]\}$.

Köbler et al. [28] proved the following theorem. The reduction from GA to UniqueGA in [28] is described in its proof.

Theorem 5. [28–Theorem 1.31] GA is polynomial-time Turing reducible to UniqueGA.

Proof. Given an oracle $\mathcal{O}$ for UniqueGA, the following algorithm solves GA in polynomial time. Let G be any given instance of GA.

(U1) Repeat (U2)-(U3) for each i starting with n down to 1.
(U2) Repeat (U3) for each j ranging from $i + 1$ to n.
(U3) Invoke $\mathcal{O}$ with input graph $G_{[1,...,i-1,i]} \cup G_{[1,...,i-1,j]}$. If the outcome of $\mathcal{O}$ is YES, output YES and halt.
(U4) Output NO.

If G is an "YES" instance, there is at least one non-trivial automorphism. Take the largest number $i \in \{1, ..., n\}$ such that there exists a number $j \in \{1, ..., n\}$ and a non-trivial automorphism $\pi \in Aut(G)_{[1,...,i]}$ for which $\pi(i) = j$ and $i \neq j$. We claim that there is exactly one such non-trivial automorphism. This is seen as follows. First, note that $Aut(G)_{[1,...,i-1]}$ is expressed as $Aut(G)_{[1,...,i-1]} = \pi_1 Aut(G)_{[1,...,i]} + \cdots + \pi_d Aut(G)_{[1,...,i]}$. For any two distinct cosets $\pi_s Aut(G)_{[1,...,i]}$ and $\pi_t Aut(G)_{[1,...,i]}$ and for any two automorphisms $\sigma \in \pi_s Aut(G)_{[1,...,i]}$ and $\sigma' \in \pi_t Aut(G)_{[1,...,i]}$, it holds that $\sigma(i) \neq \sigma'(i)$. Since $|Aut(G)_{[1,...,i]}| = 1$ and there exists the unique coset $\pi_k Aut(G)$ such that $\sigma(i) = j$ for any $\sigma \in \pi_k Aut(G)$ by the definition of i, we obtain $|\pi_k Aut(G)_{[1,...,i]}| = 1$. This implies that the non-trivial automorphism π is unique. Note that the unique non-trivial automorphism interchanges two subgraphs $G_{[1,...,i-1,i]}$ and $G_{[1,...,i-1,j]}$. Therefore, the above algorithm successfully outputs YES at Step (U3).

On the contrary, if G is a "NO" instance, then for every distinct i and j, the modified graph has no non-trivial automorphism. Thus, the above algorithm correctly rejects such a graph G. □

Finally, we describe the reduction from GA to UniqueGA$_{ff}$ by slightly modifying the reduction given in the above proof.

Lemma 6. GA is polynomial-time Turing reducible to UniqueGA$_{ff}$.

Proof. We only need to change the number of nodes to invoke oracle UniqueGA$_{ff}$ in (U3). To do so, we first modify the size of each label. Since the number m of all nodes $G_{[1,...,i-1,i]} \cup G_{[1,...,i-1,j]}$ is even, if there is no k such that $m = 2(2k + 1)$ then we add one more node appropriately to the original labels. We then attach our modified labels of length $2n+i+4$ and $2n+j+4$ to nodes i and j, respectively. Note that this modified graph satisfies the promise of UniqueGA$_{ff}$. Our algorithm therefore works correctly for any instance of GA. □

Approximate Quantum Error-Correcting Codes and Secret Sharing Schemes

Claude Crépeau[1,⋆], Daniel Gottesman[2,⋆⋆], and Adam Smith[3,⋆⋆⋆]

[1] McGill University, Montréal, QC, Canada
crepeau@cs.mcgill.ca
[2] Perimeter Institute, Waterloo, ON, Canada
dgottesman@perimeterinstitute.ca
[3] Weizmann Institute of Science, Rehovot, Israel
adam.smith@weizmann.ac.il

Abstract. It is a standard result in the theory of quantum error-correcting codes that no code of length n can fix more than $n/4$ arbitrary errors, regardless of the dimension of the coding and encoded Hilbert spaces. However, this bound only applies to codes which recover the message exactly. Naively, one might expect that correcting errors to very high fidelity would only allow small violations of this bound. This intuition is incorrect: in this paper we describe quantum error-correcting codes capable of correcting up to $\lfloor (n-1)/2 \rfloor$ arbitrary errors with fidelity exponentially close to 1, at the price of increasing the size of the registers (i.e., the coding alphabet). This demonstrates a sharp distinction between exact and approximate quantum error correction. The codes have the property that any t components reveal no information about the message, and so they can also be viewed as error-tolerant secret sharing schemes.

The construction has several interesting implications for cryptography and quantum information theory. First, it suggests that secret sharing is a better classical analogue to quantum error correction than is classical error correction. Second, it highlights an error in a purported proof that verifiable quantum secret sharing (VQSS) is impossible when the number of cheaters t is $n/4$. In particular, the construction directly yields an *honest-dealer* VQSS scheme for $t = \lfloor (n-1)/2 \rfloor$. We believe the codes could also potentially lead to improved protocols for dishonest-dealer VQSS and secure multi-party quantum computation.

More generally, the construction illustrates a difference between exact and approximate requirements in quantum cryptography and (yet again) the delicacy of security proofs and impossibility results in the quantum model.

⋆ Supported in part by Québec's MDER, FQRNT, Canada's NSERC, MITACS, CIAR and the Bell University Laboratories. Some of this research was done while the author was visiting MSRI, Berkeley CA.

⋆⋆ Some of this research was done while the author was supported by the Clay Mathematics Institute, and some while the author was visiting MSRI, Berkeley CA.

⋆⋆⋆ Some of this research was done while the author was a student at the MIT CSAIL.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 285–301, 2005.

1 Introduction

Quantum computers are likely to be highly susceptible to errors from a variety of sources, much more so than classical computers. Therefore, the study of quantum error correction is vital not only to the task of quantum communications but also to building functional quantum computers. In addition, quantum error correction has many applications to quantum cryptography. For instance, there is a strong connection between quantum error-correcting codes and secret sharing schemes [6], and that connection was combined with fault-tolerant quantum computation to perform multiparty secure quantum computations [9]. Many quantum key distribution schemes also rely on ideas from quantum error-correction for their proofs of security. Thus, bounds on the performance of quantum error-correcting codes (QECCs) in various scenarios are relevant both to the foundations of quantum information theory and to quantum cryptography.

It is an immediate result of the no-cloning theorem [24] that no quantum error-correcting code of length n can fix $n/2$ erasures: such a code would allow one to reconstruct two copies of an encoded quantum state from two halves of the full codeword, which would be cloning the state. This result is valid regardless of the dimension of the coding Hilbert space. Another well known result from the theory of quantum error correction is that a length n code can fix t arbitrary single position errors if and only if it can fix $2t$ erasure errors [11]. This follows immediately from the quantum error-correction conditions [11]

$$\langle \psi_i | E_a^\dagger E_b | \psi_j \rangle = C_{ab} \delta_{ij} \tag{1}$$

(for basis encoded states $\{|\psi_i\rangle\}$ and correctable errors $\{E_a\}$) and implies that no QECC of length n can fix more than $n/4$ arbitrary errors, regardless of the dimension of the coding and encoded Hilbert spaces. In contrast, a classical repetition code can correct up to $\lfloor (n-1)/2 \rfloor$ errors.

In this paper, we describe QECCs of length n that can correct arbitrary errors which affect up to $t = \lfloor (n-1)/2 \rfloor$ positions, with the guarantee that the fidelity of the reconstructed state will be exponentially close to 1. That is, *approximate* quantum error-correcting codes have the capability of correcting errors in a regime where no *exact* QECC will function. The scheme is also a secret-sharing scheme, in that no t positions reveal any information at all about the message. The result has a number of implications for both cryptography and quantum information theory:

- It may be possible to build approximate QECCs which are highly efficient and yet useful in common error correction scenarios, improving on exact QECCs for the same scenarios. In most cases, exact reconstruction of the quantum state is not necessary, so a more efficient approximate QECC would be welcome.
- The connection between correcting general errors and erasure errors breaks down for approximate QECCs. This suggests there is no sensible notion of distance for an approximate quantum error-correcting code.

- The proof of the impossibility of *verifiable quantum secret sharing* (VQSS) with $t \geq n/4$ cheaters in [9] is incorrect, since it assumes that the $t < n/4$ bound on error correction extends to approximate quantum codes. In particular, the construction described here immediately yields an honest-dealer verifiable quantum secret sharing scheme which is secure for $t = \lfloor (n-1)/2 \rfloor$. Similar constructions may allow verifiable quantum secret sharing (VQSS) with a *dishonest* dealer and secure multiparty quantum computation (MPQC) beyond previously known bounds. We have devised candidate protocols for these tasks allowing up to $(n-1)/2$ cheaters, but we do not present them here, as we have not yet proved their security and they are, in any case, quite complex.
- Secret sharing may serve as a better classical analogue to quantum error correction than does classical error correction. The sharp difference we see between perfect and approximate quantum error correction parallels to some extent a similar difference between error-tolerant secret sharing schemes (explained below) with zero error and those with exponentially small error [19]. The codes here use such secret sharing schemes as a building block.
- More generally, our results demonstrate that there can be a dramatic difference in behavior between the exact performance of some quantum-mechanical task and approximate performance of the task, even when the approximation is exponentially good. A similar divergence between exact and approximate bounds has recently been seen in the context of private quantum channels [13]. These examples serve as a caution — especially valid in cryptography — that intuition about approximate performance of quantum protocols may be misleading.

The idea of using a randomized encoding algorithm is not new in QECC. In particular [4] have devised codes that can correct more (malicious) errors on average than any deterministic QECC. However, their model significantly differs from ours in one of two ways: they assume either that the errors occur at random or that the code is randomly agreed on by the coder and the decoder but is kept secret from the adversarial noise source. This model does not seem suitable in cryptographic applications such as VQSS and MPQC [9]. In our model no secret is shared by the coder and decoder. However, part of our code can be viewed as providing a way for the coder to information-theoretically encrypt the necessary secret. (This is possible since the adversary only has access to part of the transmitted state, though it could be any part.)

A closer analogue to our codes is present in [15], which gave a pure-state encoding to approximately correct a specific error model more efficiently than a typical minimum-distance code. (Note, however, that the nature of the error model in fact precludes *any* exact quantum error-correcting code.) Closer yet is [21], which considered approximate quantum error correction in precisely our sense, and studied conditions for approximate error correction to be possible. They did not, however, present any specific codes or suggest that approximate QECCs might allow significant improvements in the number of correctable registers.

Secret Sharing and Quantum Error Correction. Classically, an (n, d)-secret sharing scheme splits a secret into n pieces so that no $d-1$ shares reveal any information about the secret, but any d shares allow one to reconstruct it. Such a scheme is already an error-correcting code, since it allows one to correct up to $n-d$ erasures. Error-correcting codes need not be secret sharing schemes: a repetition code, for example, provides no secrecy at all. In the quantum world, the connection is much tighter. Cleve *et al.* [6] observed that any (perfect) QECC correcting t erasures is itself a secret sharing scheme, in that no t components of the code reveal any information about the message. This follows from the principle that information implies disturbance. Furthermore, most known (perfect) classical secret sharing schemes (and "ramp" schemes) can be directly transformed into (perfect) QECC's with the related parameters [22].

The quantum code construction described here illustrates a further connection to classical secret sharing. An error-tolerant secret sharing scheme (ETSS) can recover the secret even when t shares have been maliciously corrupted. Ordinary (n, d)-secret sharing schemes are error-tolerant: such a scheme corrects $n-d$ erasures and hence $t = (n-d)/2$ errors (this fact was first highlighted for Shamir secret sharing in [16]). If we also want any t shares to reveal no information, then we get $t < d$, and thus $t < n/3$. This is optimal for schemes with zero error probability. On the other hand, if one allows a small probability of mistaken error correction, then one can in fact get error-tolerant secret sharing schemes which correct $t = \lfloor (n-1)/2 \rfloor$ errors (see the Preliminaries for more details). Thus, the best classical analogue for approximate quantum codes are error-tolerant classical secret sharing schemes which correct any t errors with high probability. These have been studied more or less explicitly in work on multi-party computation [19, 7, 8].

It is worth noting that the construction of quantum error-tolerant secret sharing schemes has farther reaching implications than analogous classical constructions. Our approximate quantum codes correct a number of general errors for which no exact code would suffice, whereas the classical constructions can be better understood as reducing the number of erasures that can be corrected via secret sharing techniques. A straightforward classical repetition code already corrects up to $\lfloor (n-1)/2 \rfloor$ arbitrary errors exactly, so there is no need to resort to sophisticated techniques to achieve this with classical ECCs.

Results. Our construction produces quantum codes which encode ℓ qubits into n registers of $\frac{\ell}{(n-2t)} + O(ns)$ qubits each and which correct any t adversarial errors with probability 2^{-s} (the bound assumes $\log n < \ell < 2^s$ for simplicity). This is done by transforming $[[n, 1, n/2]]_n$ QECCs on n-dimensional registers into better codes on $2^{O(ns)}$-dimensional registers. The codes we construct are always decodable in polynomial time, since the only necessary operations are verification of quantum authentication and erasure correction for a stabilizer code, and since erasure correction for a stabilizer code only requires solving a system of linear equations.

2 Preliminaries

Classical Authentication. For our purposes, a classical (one-time) authentication scheme is a function $h_a(m)$ that takes a secret key a and a message m as input (and no other randomness), and outputs a tag for the message. Typically, Alice sends the pair $m, h_a(m)$ to Bob, with whom she shares the key a. Bob receives a pair m', tag' and accepts the message as valid if and only if $tag' = h_a(m')$. Bob will always accept a message that really came from Alice. The scheme has error ϵ if, given a valid pair $m, h_a(m)$, no adversary Oscar can forge a tag for a different message m' with probability better than ϵ. That is, for all messages m and all (computationally-unbounded, randomized) algorithms $O()$, if a is chosen randomly from a set of keys $\mathcal{K}$, then:

$$\Pr_{a \leftarrow \mathcal{K}}[m', tag' \leftarrow O(m, h_a(m)) \ : \ tag' = h_a(m')] \leq \epsilon.$$

We make no assumptions on the running time of the adversary. If the message is ℓ bits long, then one can find a polynomial time authentication scheme where both the key and the tags have length $O(\log \ell + \log(\frac{1}{\epsilon}))$ (see, e.g., [10]).

For the remainder of this paper, we assume the reader is familiar with the basic notions and notation of quantum computing (see a textbook such as [17] if necessary).

Quantum Authentication. Intuitively, a quantum authentication scheme [2] is a keyed system which allows Alice to send a state ρ to Bob with a guarantee: if Bob accepts the received state as "valid", the fidelity of that state to ρ is almost 1. Moreover, if the adversary makes no changes, Bob always accepts and the fidelity is exactly 1. The following definition is from Barnum et al. [2]. We first define what constitutes a quantum authentication scheme, and then give a definition of security.

Definition 1 ([2]). *A* quantum authentication scheme *(*QAS*) is a pair of polynomial time quantum algorithms A and V together with a set of* classical *keys $\mathcal{K}$ such that:*

- *A takes as input an m-qubit message system M and a key $k \in \mathcal{K}$ and outputs a transmitted system C of $m + t$ qubits.*
- *V takes as input the (possibly altered) transmitted system $\hat{C}$ and a classical key $k \in \mathcal{K}$ and outputs two systems: a m-qubit message state $\hat{M}$, and a single (verdict) qubit V which indicates acceptance or rejection. The classical basis states of V are called $|\text{ACC}\rangle, |\text{REJ}\rangle$ by convention.*

For any fixed key k, we denote the corresponding super-operators by A_k and V_k.

Bob may measure the qubit V to see whether or not the transmission was accepted or rejected. Nonetheless, we think of V as a qubit rather than a classical bit since it will allow us to describe the joint state of the two systems $\hat{M}, V$ with a density matrix. Given a pure state $|\psi\rangle \in \mathcal{H}_M$, consider the following test on

the joint system $\hat{M}, V$: output a 1 if the first m qubits are in state $|\psi\rangle$ *or* if the last qubit is in state $|\text{REJ}\rangle$ (otherwise, output a 0). The projectors corresponding to this measurement are

$$P_1^{|\psi\rangle} = |\psi\rangle\langle\psi| \otimes |\text{ACC}\rangle\langle\text{ACC}| \ + \ I_{\hat{M}} \otimes |\text{REJ}\rangle\langle\text{REJ}|$$
$$P_0^{|\psi\rangle} = (I_{\hat{M}} - |\psi\rangle\langle\psi|) \otimes (|\text{ACC}\rangle\langle\text{ACC}|)$$

We want that for all possible input states $|\psi\rangle$ and for all possible interventions by the adversary, the expected fidelity of V's output to the space defined by $P_1^{|\psi\rangle}$ is high. This is captured in the following definition of security.

Definition 2 ([2]). *A* QAS *is secure with error ϵ for a state $|\psi\rangle$ if it satisfies:*

- Completeness: *For all keys $k \in \mathcal{K}$: $V_k(A_k(|\psi\rangle\langle\psi|)) = |\psi\rangle\langle\psi| \otimes |\text{ACC}\rangle\langle\text{ACC}|$*
- Soundness: *For a super-operator $\mathcal{O}$, let ρ_{Bob} be the state output by Bob when the adversary's intervention is characterized by $\mathcal{O}$, that is: $\rho_{Bob} = \frac{1}{|\mathcal{K}|}\sum_k V_k(\mathcal{O}(A_k(|\psi\rangle\langle\psi|)))$ (this is the expectation over all values of the key of the state output by Bob). The* QAS *has soundness error ϵ for $|\psi\rangle$ if for all super-operators $\mathcal{O}$,*

$$\text{Tr}\left(P_1^{|\psi\rangle}\rho_{Bob}\right) \geq 1 - \epsilon$$

A QAS *is secure with error ϵ if it is secure with error ϵ for all states $|\psi\rangle$. We make no assumptions on the running time of the adversary.*

In order to authenticate a message of ℓ qubits, the authentication scheme of [2] uses a (classical) key of length $2\ell + O(\log(\frac{1}{\epsilon}))$ random bits and produces a transmitted system of $\ell + O(\log(\frac{1}{\epsilon}))$ qubits. The large part 2ℓ of the classical key is used to the encrypt the quantum state, which is necessary for any quantum authentication scheme to be secure [2]. In the special case where Alice wishes to authenticate half of a maximally entangled state $\sum |i\rangle|i\rangle$, in fact only $O(\log(\frac{1}{\epsilon}))$ classical key bits are necessarily [18, 12], effectively because Alice's message is already a maximally mixed state, making encryption redundant.

Composability of Quantum Authentication. We will need authentication protocols that have an additional composability property: If (A_k, V_k) is a QAS with error ϵ for key k, then the concatenated protocol

$$\left(\bigotimes_{i=1}^{n} A_{k_i}, \bigotimes_{i=1}^{n} V_{k_i}\right) \tag{2}$$

should be a QAS with error ϵ for the key $(k_1, \ldots, k_n)$, with the understanding that the concatenated verification protocol accepts if and only if all of the tensor components accept (i.e. the verdict qubit for the concatenated scheme is the logical AND of the individual verdict qubits).

This sort of composability holds trivially for a classical authentication scheme, although the error may increase linearly with the number of compositions. We

do not know if the same is true in general for quantum authentication schemes. However, the quantum authentication schemes of [2] are indeed composable, with no blow-up in the error parameter. This follows because they are constructed from stabilizer purity testing codes (PTCs), which clearly satisfy a corresponding property (if Q_k is a stabilizer PTC with error ϵ, then $\bigotimes_{i=1}^{n} Q_{k_i}$ is a stabilizer PTC with error ϵ).

Classical Secret Sharing and Error Correction. A classical (n, d)-secret sharing scheme [20] is a cryptographic protocol allowing a *dealer* to share a secret k into n *shares* $(s_1, \ldots, s_n)$ with n *share-holders* $P_1, \ldots, P_n$ in such a way that any $d-1$ s_i's contains no information about k whereas any d of those s_i's completely define k. We write $(s_1, \ldots, s_n) \in_R SS_{n,d}(k)$, a random instantiation of a set of shares for secret k. The original construction of Shamir [20], based on Reed-Solomon codes, allows one to share an ℓ-bit secret with shares that are each $\max\{\ell, \log n\}$ bits.

An important component in our construction is a classical secret sharing scheme which allows the honest players to reconstruct the secret even if the cheaters alter their shares. Specifically, consider the following game: an honest dealer takes a secret, splits it into n shares $s_1, .., s_n$, and distributes the shares amongst n participants over secure channels (i.e., player i gets only s_i). Next, an adversary (adaptively) corrupts up to $t = d-1$ of the players. Finally, all players send their (possibly corrupted) shares over secure channels to a trusted arbiter who attempts to recover the secret. The secret sharing scheme is called an *error-tolerant secret sharing scheme* (ETSS) and is *t-error-correcting with error ϵ* if the arbiter can reconstruct the correct secret with probability $1 - \epsilon$, regardless of the adversary's strategy. In other words, an ETSS is a secret-sharing scheme which also acts as an error-correcting code correcting any t errors with high probability.

Error-tolerant secret sharing has been studied under the names "honest-dealer VSS with a non-rushing adversary" [8] and "non-interactive Las Vegas perfectly secure message transmission" [23]. "Robust secret sharing" [5] is a slightly weaker variant of the problem. Another variant, "honest-dealer VSS with rushing" is slightly stronger than ETSS; see [8] for a discussion of the differences.

A number of constructions of ETSS schemes appear in the literature. When $t < n/3$, any ordinary secret sharing scheme is in fact an ETSS with zero error (since it is a code correcting $2t$ erasures and hence t errors). This connection was first pointed out by [16]. When t is between $n/3$ and $n/2$, one can adapt constructions from multi-party computation protocols [19, 7, 8]. We will use a simple construction for the case $t = \lfloor (n-1)/2 \rfloor$ from [8]. The dealer encodes the secret using an ordinary secret sharing scheme, and augments the shares by creating a fresh authentication key and tag for every pair of players: P_i gets the key a_{ij} and P_j gets the tag $h_{a_{ij}}(s_j)$. If the adversary does not succesfully forge any authentication tags for keys held by honest players, then the arbiter can reconstruct the secret by accepting only shares for which at least $t + 1$ of the authentication tags are valid.

The two schemes suggested above tolerate the maximum number of cheaters. On one hand, schemes with zero error can tolerate at most $n/3$ errors [19]. On the other hand, it is clear that no ETSS scheme can correct more than $t = \lfloor (n-1)/2 \rfloor$ errors: any $n-t$ players must be able to reconstruct the secret alone (as the adversary could simply erase all its shares), and so we must have $n-t > t$. Alternatively, one can view this as an ordinary error correction bound: if the adversary could control half of the shares, he could make them all consistent with a value of his choosing (say 0) and force the arbiter to reconstruct 0.

The main complexity measure of an ETSS scheme is the share size. For a given scheme, let $CC(\ell, \epsilon, t)$ denote the maximum size (in bits) of a share held by any player. When $t < n/3$, the usual Shamir secret sharing scheme is a zero-error ETSS scheme with zero error and share size $CC(\ell, 0, t) = \ell/(n-3t)$ (for $\ell > (n-3t)\log n$). The errors can be corrected in polynomial time since the scheme encodes data in a Reed-Solomon code. For $t = \lfloor (n-1)/2 \rfloor$, the augmented scheme using authentication tags produces shares of size $CC(\ell) = \ell + O(n \log(\frac{1}{\epsilon}))$ (when $\ell > \log n$ and $\log(\frac{1}{\epsilon}) > \max\{n, \ell\}$).

Based on [5], Cramer et al. [8] present a more compact scheme for $t = \lfloor (n-1)/2 \rfloor$ with share size $O(\ell + n + \log(\frac{1}{\epsilon}))$. Unfortunately, that scheme is not known to correct the errors in polynomial time. A second scheme, for t further away from $n/2$, generates shares of size $CC(\ell, \epsilon, t) = \Omega(n \log(\frac{1}{\epsilon}) + \ell/(n-2t))$. The same work [8] also proved a simple lower bound on the share size of ETSS schemes: $CC(\ell, \epsilon, t) = \Omega(\log(\frac{1}{\epsilon}) + \frac{\ell}{(n-2t)})$. This bound is tight for $\log(\frac{1}{\epsilon}) > n$ and $n = 2t+1$.

3 Definition of Approximate Quantum Codes (AQECC)

An approximate quantum error-correcting code allows Alice to send a state ρ to Bob with the guarantee that if few enough errors occur in transmission, the fidelity of the state received by Bob to ρ will be almost 1.

Let $q = p^m$ and $Q = p^N$ for some prime p and integers m, N. We first define what constitutes an AQECC over $\mathbb{F}_Q$, and then give a definition of correctness. (Note that the definition makes sense over any alphabet, but we restrict to prime powers for simplicity).

Definition 3. *An* approximate quantum error correcting code *(AQECC) is a pair of quantum algorithms E (encoder) and D (decoder) such that:*

- *E takes as input a m-quqit message system M and outputs a (mixed state) codeword C of n quQits.*
- *D takes as input the (possibly altered) transmitted system $\hat{C}$ and outputs a m-quqit message state $\hat{M}$.*

In our constructions, both the encoding E and error-correction algorithm D run in polynomial time in the number of qubits of input.

We will define the correctness of an AQECC on pure states, but it follows from a result of Barnum, Knill and Nielsen ([3], Thm 2) that the output of the

AQECC also has high fidelity to an input which is mixed or part of an entangled state.

Given a pure state $|\psi\rangle \in \mathcal{H}_M$, consider the following test on the system $\hat{M}$: output a 1 if the first k quqits are in state $|\psi\rangle$ (otherwise, output a 0). The projectors corresponding to this measurement are

$$P_\psi = |\psi\rangle\langle\psi|$$
$$P_\psi^\perp = (I_{\hat{M}} - |\psi\rangle\langle\psi|)$$

We want that for all possible input states $|\psi\rangle$ and for all possible interventions by the adversary, the expected fidelity of Bob's output to the space defined by P_ψ is high. This is captured in the following definition of correctness.

Definition 4. *An AQECC is t-correct with error ϵ for a state $|\psi\rangle$ if for all super-operators $\mathcal{O}$ acting on at most t quQits (that is, $\mathcal{O}$ can be written as $I_{n-t} \otimes \tilde{\mathcal{O}}_t$ for some partition of the system into $n-t$ and t quQits),*

$$\mathrm{Tr}\,(P_\psi \rho_{Bob}) \geq 1 - \epsilon,$$

where ρ_{Bob} is the state output by Bob when the adversary's intervention[1] *is characterized by $\mathcal{O}$, that is:*

$$\rho_{Bob} = D(\mathcal{O}(E(|\psi\rangle\langle\psi|))).$$

An AQECC is t-correct with error ϵ if it is t-correct with error ϵ for all states $|\psi\rangle$.

4 A Length 3 Quantum Code Approximately Correcting One Arbitrary Error

We start with a small example, from a well known code. The code c corrects one erasure error:

$$\begin{aligned} |0\rangle &\to |000\rangle + |111\rangle + |222\rangle \\ |1\rangle &\to |012\rangle + |120\rangle + |201\rangle \\ |2\rangle &\to |021\rangle + |102\rangle + |210\rangle \end{aligned} \tag{3}$$

Let $H_1 \otimes H_2 \otimes H_3$ be the coding space of the original code

$$c|\psi\rangle \in H_1 \otimes H_2 \otimes H_3,$$

and let (A_k, V_k) be a quantum authentication scheme as constructed in [2].

[1] We make no assumptions on the running time of the adversary.

We construct a three-component code c' as follows:

$$\begin{aligned} c'|\psi\rangle = &(A_{k_1}(H_1), k_2, k_3), \\ &(A_{k_2}(H_2), k_1, k_3), \\ &(A_{k_3}(H_3), k_2, k_1). \end{aligned} \tag{4}$$

Let $H'_1 \otimes H'_2 \otimes H'_3$ be the coding space of the new code

$$c'|\psi\rangle \in H'_1 \otimes H'_2 \otimes H'_3$$

Note that k_1, k_2, and k_3 are random *classical* strings which we use as keys for the quantum authentication protocol A_k. Thus, the H'_is contain both quantum and classical information. Intuitively, we use the QAS to ensure that an adversary cannot change the quantum state of a single register without being detected; thus, we can transform general errors into erasure errors, allowing us to correct one faulty register out of three (no exact QECC can do this). Then we distribute the authentication keys among the three registers so that Bob can recover them. We must, however, do so in a way that prevents an adversary with access to a single register from either learning the key applying to her own register (which would allow her to change the quantum state) or from preventing reconstruction of the classical keys.

Theorem 1. *If A_k is a* QAS *secure with error ϵ then c' is a* 1*-correct AQECC with error prob.* $\mathrm{poly}(\epsilon)$*, correcting one arbitrary error.*

We omit the proof of this theorem, as in Section 5 we will prove a more general result.

4.1 Reconstruction

In all cases, the reconstruction has two phases. First we reconstruct the classical keys and use them to verify and decode the quantum authentications. This may result in discarding one register, but at least two remain, which is enough for the erasure-correcting code to recover the original encoded state. Consider the following cases:

- All k_i's agree in H'_1, H'_2, H'_3:
  Recover k_i from either $H'_j, j \neq i$, check that $A_{k_i}(H_i)$ properly authenticates H_i. If one authentication fails, ignore the improperly authenticated H_i and reconstruct the valid codeword as $c|\psi\rangle \in H_1 \otimes H_2 \otimes H_3$ using the erasure recovery algorithm from both $H_j, j \neq i$.
- Some H'_i disagrees with H'_j, H'_h on both keys k_h and k_j:
  Discard register i, which must be corrupted. Recover k_j from H'_h and k_h from H'_j, and decode the authentications $A_{k_j}(H_j)$ and $A_{k_h}(H_h)$ (which should both pass, since only one register can fail). Reconstruct the valid codeword as $c|\psi\rangle \in H_1 \otimes H_2 \otimes H_3$ using the erasure recovery algorithm from H_j and H_h.

- H'_i and H'_j disagree on key k_h, while H'_h agrees with everyone:
  Either register i or j is corrupt. Get k_i and k_j from H'_h and check that $A_{k_i}(H_i)$ properly authenticates H_i, and that $A_{k_j}(H_j)$ properly authenticates H_j. If neither fails, reconstruct the valid codeword as $c|\psi\rangle \in H_1 \otimes H_2 \otimes H_3$ using the erasure recovery algorithm from H_i and H_j. If one fails, say $A_{k_i}(H_i)$, then conclude register i is corrupt and recover k_h from H'_j, decode $A_{k_h}(H_h)$, and reconstruct the valid codeword as $c|\psi\rangle \in H_1 \otimes H_2 \otimes H_3$ using the erasure recovery algorithm from H_h and H_j.

Other cases cannot arise, since only one register can have been changed from the original encoding.

5 A General n-Component Approximate QECC Family Correcting up to $d-1 < n/2$ Arbitrary Errors

In order to generalize the above construction to cases with n registers, we need to systemize the distribution of the classical keys. Again, it is helpful to imagine that we are trying to defeat an adversary with access to $t < n/2$ components of the code. Recall that we needed two conditions: First, the adversary should not be able to learn the classical key for her register, but the receiver Bob should be able to reconstruct the keys. Second, the adversary should not be able to interfere with Bob's reconstruction of the keys.

These are precisely the properties of an ETSS. This suggests the following strategy for building a t-correct AQECC: encode $|\psi\rangle$ using a distance $t+1$ QECC, authenticate the n components using keys $\boldsymbol{k} = k_1, ..., k_n$, and then share $\boldsymbol{k}$ using a classical ETSS. The result could be considered to be a quantum ETSS (that is, an ETSS for quantum data). However, the ramifications of this construction for quantum data are more far-reaching than for the classical protocol. Not only does the quantum ETSS have potential cryptographic applications, but it demonstrates the possibility of exceeding the no-cloning bound on QECCs. Indeed, any QECC, exact or approximate, is in some sense a quantum ETSS — the ability to (approximately) correct erasures on a set of registers implies that an adversary with access to those registers can gain (almost) no information about the encoded data [21].

Let $\mathcal{Q}$ be a QECC that can correct $d-1 < n/2$ arbitrary erasure errors: $\mathcal{Q} = [[n, k, d]]$. Such a code can be constructed over sufficiently large dimension Q; for instance, use a polynomial quantum code [1]. The coding space of $\mathcal{Q}$ is defined as

$$\mathcal{Q}|\psi\rangle \in H_1 \otimes H_2 \otimes H_3 \otimes ... \otimes H_n.$$

We assume $\dim(H_1) = \dim(H_2) = ... = \dim(H_n)$.

We construct a new code $\mathcal{Q}'$ over larger Hilbert spaces that can correct $d-1 < n/2$ arbitrary errors except with small probability. Register i of the n-component code $\mathcal{Q}'$ contains the following:

$$\langle A_{k_i}(H_i), s_i, [a_{ij}(\forall j \neq i)], [h_{a_{ji}}(s_i)(\forall j \neq i)]\rangle, \tag{5}$$

where we have used the classical authentication scheme (in systematic form):

$$m, a \to (m, h_a(m)), \tag{6}$$

which has error ϵ, and $(s_1, \ldots, s_n) \in_R SS_{n,d}(k_1, \ldots, k_n)$, a secret sharing scheme such that any $d-1$ s_i's contains no information about $(k_1, \ldots, k_n)$ whereas any d of those s_i's completely define $(k_1, \ldots, k_n)$. The combination of classical secret sharing and classical authentication forms an ETSS [8], as described above; in fact, any ETSS would do.

For instance, the $n = 3$ case of this construction is as follows:

$$\begin{aligned} c'|\psi\rangle = &(A_{k_1}(H_1), s_1, [a_{12}, a_{13}], [h_{a_{21}}(s_1), h_{a_{31}}(s_1)]), \\ &(A_{k_2}(H_2), s_2, [a_{21}, a_{23}], [h_{a_{12}}(s_2), h_{a_{32}}(s_2)]), \\ &(A_{k_3}(H_3), s_3, [a_{31}, a_{32}], [h_{a_{13}}(s_3), h_{a_{23}}(s_3)]). \end{aligned} \tag{7}$$

Note that this is more complicated than the scheme in section 4. Instead of giving the keys k_i to the other two players, we have instead shared them among all three players, so no single component has access to any of the three keys used for quantum authentication. In section 4, we were able to use the fact that the quantum register attacked by the adversary must be the same as the classical register attacked, so it is only necessary to protect information about one of the keys k_i, not all of them. With the extra flexibility granted the adversary by being able to attack multiple registers, it is more straightforward to protect all n keys with the classical ETSS.

We are now ready for our main result. Let $H'_1 \otimes H'_2 \otimes ... \otimes H'_n$ be the coding space of the new code

$$\mathcal{Q}'|\psi\rangle \in H'_1 \otimes H'_2 \otimes ... \otimes H'_n$$

Theorem 2. *If A_k is a* QAS *secure with error ϵ, Q is a non-degenerate stabilizer code with distance d, and $h_a(\cdot)$ is a classical authentication scheme with error ϵ, then $\mathcal{Q}'$ is an approximate quantum error-correcting code correcting $d-1$ arbitrary errors with error at most $2n^2\epsilon$.*

5.1 Reconstruction

The reconstruction procedure is similar to that for the previous protocol, but slightly more involved, since we must verify the classical authentications as well. Rather than breaking the procedure into different cases, in this version of the protocol, we can systematically go through four steps: First, verify the classical authentications and discard any invalid classical share. Second, reconstruct the keys k_i. Third, verify and decode the quantum authentications. Fourth, discard any invalid quantum register and reconstruct the encoded quantum state.

1. Verify classical authentications:
   For each s_i, consider it valid if at least half its authentications are correct according to $a_{ji}, j \neq i$. Discard any share s_i which is not valid.

2. Reconstruct the keys k_i:
   Up to $d-1$ shares s_i can have been discarded in the first stage, so at least $n-d+1 \geq n/2+1 > d$ shares remain. Use these to reconstruct $(k_1, \ldots, k_n)$. If the remaining shares are not all consistent with a single value of the secret, Bob aborts and outputs the quantum state $|0\rangle$.
3. Verify and decode the quantum authentications:
   Use the key k_i to verify and decode the quantum authentication $A_{k_i}(H_i)$.
4. Reconstruct the encoded quantum state:
   Discard any registers which failed the quantum authentication, and use the remaining registers to reconstruct the valid codeword as $c|\psi\rangle \in H_1 \otimes \ldots \otimes H_n$ using the erasure recovery algorithm. (At most $d-1$ have been discarded.) If the remaining registers are not consistent with a single quantum codeword, Bob aborts and outputs the quantum state $|0\rangle$.

We prove this assuming the original QECC $\mathcal{Q}$ is a nondegenerate CSS code (which is sufficient to demonstrate that AQECCs exist correcting up to $(n-1)/2$ errors), but the proof can easily be extended to an arbitrary stabilizer code.

Proof (of Theorem 2). If no errors occurred, the above procedure will exactly reconstruct the original encoded state. We need to show that it still approximately reconstructs the state when there are up to $d-1$ arbitrary errors in unknown locations. Let B be the set of registers attacked by the adversary, and let $A = [n] \setminus B$ be the registers held by honest players.

The intuition for the proof is simple. With high probability, the authentication keys will be reconstructed correctly; conditioned on that event, all components of the QECC which pass the authentication test should be "close" to the encoding of $|\psi\rangle$ restricted to those positions, and applying erasure correction should yield a state very close to $|\psi\rangle$. Formalizing this intuition is more delicate than it would be if the data involved were classical. The quantum version of the statement "such-and-such event holds with probability $1-\epsilon$" is "the state of the system has fidelity at least $1-\epsilon$ to the subspace such-and-such." The problem lies in the fact that the union bound from ordinary probability, which is the basis of the intuition outlined above, does not always hold in the quantum world. Our solution follows the lines of the "quantum to classical reductions" in [14, 9]. We define a set of "target" subspaces whose projectors commute (in other words, there exists a single basis of the state space in which all the projectors are diagonal), and show that the system lies close to each of these target subspaces. For commuting subspaces, the union bound does hold: if the system has high fidelity to each of the subspaces, then in fact it has high fidelity to their intersection. To complete the proof it is sufficient to show that for states in the intersection, the initial input $|\psi\rangle$ is reconstructed exactly.

The first step is to take care of the classical component of the encoding (composed of the shares s_i, classical authentication keys a_{ij} and tags $h_{a_{ij}}(s_j)$). We rely on three observations. First, we may assume w.l.o.g. that the recovery procedure measures all the classical components in the computational basis before doing any processing; thus, the state received by the reconstructor Bob

is a mixture (not a superposition) over different bit strings which he might be sent instead of the original ones. Second, the classical information held by the adversary is statistically independent of $\boldsymbol{k} = (k_1, ..., k_n)$, the vector of quantum authentication keys. (This follows from the fact that any t of the shares $s_1, ..., s_n$ are independent of the shared secret.) Third, any classical authentication tags changed by the adversary will be rejected by Bob with probability at least $1 - \epsilon$.

We define our first target subspace S_0 by the statement "the keys $\boldsymbol{k}$ reconstructed by Bob are equal to the original keys." This statement can fail only if some tag changed by the adversary is accepted by Bob, and by a (classical) union bound this can occur with probability at most $tn\epsilon < n^2\epsilon$. The fidelity to S_0 is thus at least $1 - n^2\epsilon$.

We now look at what happens within the subspace S_0. Consider the following set of measurements which might be performed by Bob after verifying the authentications, but before applying erasure correction to the code. We assume for simplicity that the adversary holds the wires $B = \{1, ..., t\}$, and the wires $A = \{t + 1, ..., n\}$ are untouched.

- For each register $i \in [n]$, $|\text{REJ}_i\rangle\langle\text{REJ}_i|$ measures whether or not Bob rejected the authentication of the i-th quantum system (correspondingly, $|\text{ACC}_i\rangle\langle\text{ACC}_i|$ measures whether or not Bob accepts).
- We use the fact that the quantum error-correcting code is a nondegenerate CSS code. The code can be defined by a sequence of parity checks performed in two bases: the standard computational basis and the rotated Fourier (or "diagonal") basis. We assume there are r independent parity checks in the rotated basis and s independent parity checks in the standard basis. Denote by V the linear space of parity checks satisfied in the computational basis, and by W the corresponding set for the Fourier basis. If the QECC code has distance at least $t + 1$, then there is a basis $v_1, ..., v_s$ of V such that, for any $i \in B$, position i is only in the support of v_i. Same for W: there is a basis of parity checks $w_1, ..., w_r$ such that only w_i involves the i-th component of the code for $i \in B$. We denote by Π_{v_i}, Π_{w_i} the corresponding projectors (that is, Π_{v_i} preserves the supspace in which the parity check v_i is satisfied).

The sets of projectors $\{|\text{REJ}_i\rangle\langle\text{REJ}_i|\}_{i\in[n]}$, $\{\Pi_{v_i}\}_{i\in[s]}$ and $\{\Pi_{w_i}\}_{i\in[r]}$ all commute with each other. The only possible interaction comes from the fact that the operators $\{\Pi_{v_i}\}$ and $\{\Pi_{w_i}\}$ operate on the same space, but they commute by definition of CSS codes. We may ignore projectors with indices $i > t$ since they correspond to checks which will always be passed within the subspace S_0: Therefore the system will have fidelity 1 to the subspaces defined by $\{\Pi_{v_i}\}$ and $\{\Pi_{w_i}\}$ for $i > t$.

We would like to claim that, whenever Bob accepts the set R of registers, R satisfies all the parity checks restricted to R. We can quantify this as follows: for all i between 1 and t, the system should lie in the subspace defined by

$$P_i = (\Pi_{v_i}\Pi_{w_i} \otimes |\text{ACC}_i\rangle\langle\text{ACC}_i|) + (I \otimes |\text{REJ}_i\rangle\langle\text{REJ}_i|). \tag{8}$$

where I is the identity operator. The security of the quantum authentication scheme, and the fact that the adversary doesn't learn anything about the keys

from the classical secret sharing, imply that the fidelity to each of these subspaces is at least $1-\epsilon$ (note: this requires the quantum authentication scheme to be secure even when composed up to t times). For $1 \leq i \leq t$, we can define the subspaces $S_1, \ldots, S_t$ corresponding to the projectors $P_1, \ldots, P_t$. By a union bound, the state of the whole system has fidelity at least $1 - n^2\epsilon - t\epsilon$ to the intersection $S = \bigcap_{i=o}^{t} S_i$. In words, S is the space of states for which Bob reconstructs the correct authentication keys, and for which the set of registers accepted by Bob satisfies all the parity checks restricted to that set.

It remains to prove that within the space S, Bob will always recover the input state $|\psi\rangle$ exactly. We may assume w.l.o.g. that Bob will measure all n of the registers which indicate whether the authentication failed or not in the basis $\{|\text{REJ}\rangle, |\text{ACC}\rangle\}$. Thus, the global state may be seen as a mixture over possible sets of registers accepted by Bob. If Bob also performs the measurements P_i, he will, with probability at least $1 - n^2\epsilon - t\epsilon$, find that the state actually satisfies all parity checks restricted to the set R of registers he accepts.

When this occurs, it then follows that applying erasure correction to R yields the same result as if we had used only registers untouched by the adversary. For a detailed proof of this fact, we refer the reader to Proposition 2.2 in [9]. The intuition behind it is straightforward: Suppose s registers are discarded, leaving up to $t-s$ registers attacked by the adversary. But because $s + (t-s) < d$, the QECC can both correct s erasures and detect an additional $t-s$ errors, so the adversary is unable to reach any state in S except the correct input state $|\psi\rangle$. We can conclude that Bob recovers a state ρ with fidelity at least $1 - 2n^2\epsilon$ to ψ, as desired.

5.2 Specific Constructions and Parameters

As mentioned above, it is natural to instantiate our construction using the polynomial codes (quantum Reed-Solomon codes) of Aharonov and Ben-Or [1]. These are nondegenerate CSS codes over an alphabet of size q whenever q is a prime power and greater than $n-1$. For any $t < n/2$, one can find a $[[n, n-2t, t+1]]_q$ code (i.e. which encodes $(n-2t)\log q$ qubits and has distance $t+1$). This means that to encode $\ell > n$ qubits, each component of the code will consist of $\ell/(n-2t)$ qubits. The components of the approximate QECC then consist of $\ell/(n-2t) + O(\log(\frac{1}{\epsilon}))$ qubits and $CC(2\ell/(n-2t) + O(\log(\frac{1}{\epsilon})), \epsilon, t)$ bits (where $CC()$ is the share size of the classical ETSS).

For $2t < n-1$, we can modify the ETSS above to get shares of size $O(n \log(\frac{1}{\epsilon}))$ $+ \ell/(n-2t)$. Putting these constructions together, we can get quantum codes where each register contains $O(n(\ell/(n-2t) + \log(\frac{1}{\epsilon})))$ qubits.

An immediate improvement can be made to these parameters by noting that, for any distance d nondegenerate stabilizer code, including the polynomial codes used here, the state of any $d-1$ registers is maximally entangled with the remaining registers. Therefore, as noted in section 2, a much shorter classical key suffices for quantum authentication. In particular, a classical key of length $O(\log \ell + \log(\frac{1}{\epsilon}))$ is sufficient to authenticate ℓ EPR halves. This leads to an approximate quantum code where each component consists of $\ell/(n-2t) + O(\log(\frac{1}{\epsilon}))$ qubits and $CC(n \log(\frac{1}{\epsilon}),$ $\epsilon, t)$ bits (when $\epsilon < 1/\ell$). This gives a total size of $\ell/(n-2t) + O(n \log(\frac{1}{\epsilon}))$.

Corollary 3 (to Theorem 2). *For $t < n/2$, there exists an approximate QECC correcting any t errors with error ϵ, where each component consists of $O(\ell/(n-2t) + n\log(\frac{1}{\epsilon})))$ qubits. When $n = 2t+1$, we get components of size $O(\ell + n\log(\frac{1}{\epsilon}))$.*

6 Discussion and Open Questions

We have constructed quantum error correcting codes that are capable of correcting general errors when up to half the registers are affected. This contrasts considerably with known upper bounds that limit a QECC to correcting errors on less than one-fourth of all registers. The price for being able to violate this bound is that we only correct the state approximately; however, we do so with exponentially good fidelity.

In general, extrapolating from exact performance of a quantum task to approximate performance is dangerous, but possible. Factors of the dimension may arise, and since the dimension is exponential in the number of qubits, dramatically different behavior becomes possible. This phenomenon is likely behind the performance of our codes, and suggests that high-fidelity AQECCs are only possible when working in high dimension.

Our codes instead consist of a small logical subspace and large registers containing both quantum and classical information. As such, they are not so useful for practical problems in quantum error correction, but do serve as an interesting in-principle demonstration of the potential power of approximate error correction. In addition, they act as quantum ETSS schemes, and may be a useful stepping stone towards building VQSS and MPQC with a large number of cheaters. Any such construction must be more complex, however, to take account of dishonest senders and receivers, and to allow the participants in the protocol to alter a state in the correct way without altering it in any unapproved manner. Indeed, it remains possible that the prior bound of $n/4$ cheaters does in fact restrict VQSS and MPQC; however, we have shown here that the existing proof of that bound does not apply to VQSS and MPQC protocols which only guarantee approximate reconstruction of the quantum state.

References

1. D. Aharonov and M. Ben-Or. Fault tolerant quantum computation with constant error rate. Preliminary version in *Proc. 29th ACM Symp. on Theory of Computing*, 1997. Submitted to SIAM J. Comp., June 1999.
2. H. Barnum, C. Crépeau, D. Gottesman, A. Tapp, and A. Smith. Authentication of quantum messages. In proceedings of The 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'02), November 16 - 19, 2002 Vancouver, BC, Canada, pages 449–458. Also Quantum Physics, abstract quant-ph/0205128, 22 pages, May 2002.
3. H. Barnum, E. Knill and M. A. Nielsen, "On Quantum Fidelities and Channel Capacities," quant-ph/980901, IEEE Trans.Info.Theor. 46 (2000) 1317-1329.

4. C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of Noisy Entanglement and Faithful Teleportation via Noisy Channels.", Phys. Rev. Lett. 76 (1996) 722-725, Quantum Physics, abstract quant-ph/9511027.
5. S. Cabello, C. Padró, G. Saéz. Secret Sharing Schemes with Detection of Cheaters for a General Access Structure. *Designs, Codes and Cryptography*, 25(2): 175-188 (2002).
6. Richard Cleve, Daniel Gottesman, Hoi-Kwong Lo. How to share a quantum secret. *Phys.Rev.Lett.* 83, p. 648–651, 1999.
7. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt and T. Rabin. Efficient Multiparty Computations Secure Against an Adaptive Adversary In *Advances in Cryptology — EUROCRYPT 1999*.
8. R. Cramer, I. Damgård, S. Fehr. On the Cost of Reconstructing a Secret. In *Advances in Cryptology — CRYPTO 2001*.
9. C. Crépeau, D. Gottesman, and A. Smith. Secure multi-party quantum computation. In Proceedings of 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada. ACM, 2002, pages 643–652.
10. P. Gemmell and M. Naor. Codes for interactive authentication. In D. R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 1994.
11. D. Gottesman, "An Introduction to Quantum Error Correction.", Quantum Physics, abstract quant-ph/0004072, 15 pages, talk given at AMS Short Course on Quantum Computation.
12. P. Hayden, D. Leung, D. Mayers. Universally composable quantum authentication. In preparation.
13. P. Hayden, D. Leung, A. Winter and P. Shor. Randomizing quantum states: Constructions and applications. *Commun. Math. Phys.* 250(2): 371-391, 2004.
14. Hoi-Kwong Lo and H. F. Chau. Unconditional security of quantum key distribution over arbitrarily long distances. *Science*, 283(5410):2050 2056, 26 March 1999.
15. D. W. Leung, M. A. Nielsen, I. L. Chuang, and Y. Yamamoto. Approximate quantum error correction can lead to better codes. Phys.Rev. A56 (1997) 2567–2573, quant-ph/9704002.
16. R. J. McEliece and D. Sarwate. On sharing secrets and Reed-Solomon codes. *Comm. ACM* 24, 1981, 583–584.
17. M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
18. J. Oppenheim and M. Horodecki. How to reuse a one-time pad and other notes on authentication, encryption and protection of quantum information. E-print quant-ph/0306161.
19. T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proc. of STOC 1989*, p. 73–85.
20. A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.
21. B. Schumacher, M. D. Westmoreland Approximate quantum error correction. E-print quant-ph/0112106, 2001.
22. A. Smith. Quantum secret sharing for general access structures. E-print quant-ph/0001087, 2000.
23. K. Srinathan, Arvind Narayanan and C. Pandu Rangan: Optimal Perfectly Secure Message Transmission. *Advances in Cryptology — CRYPTO 2004*.
24. W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned", *Nature* 299, 802, 1982 .

Compact E-Cash

Jan Camenisch[1], Susan Hohenberger[2,⋆], and Anna Lysyanskaya[3,⋆⋆]

[1] IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland
jca@zurich.ibm.com
[2] CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
srhohen@mit.edu
[3] Computer Science Department, Brown University, Providence, RI 02912, USA
anna@cs.brown.edu

Abstract. This paper presents efficient off-line anonymous e-cash schemes where a user can withdraw a wallet containing 2^ℓ coins each of which she can spend unlinkably. Our first result is a scheme, secure under the strong RSA and the y-DDHI assumptions, where the complexity of the withdrawal and spend operations is $\mathcal{O}(\ell + k)$ and the user's wallet can be stored using $\mathcal{O}(\ell + k)$ bits, where k is a security parameter. The best previously known schemes require at least one of these complexities to be $\mathcal{O}(2^\ell \cdot k)$. In fact, compared to previous e-cash schemes, our whole wallet of 2^ℓ coins has about the same size as *one* coin in these schemes. Our scheme also offers exculpability of users, that is, the bank can prove to third parties that a user has double-spent. We then extend our scheme to our second result, the first e-cash scheme that provides traceable coins without a trusted third party. That is, once a user has double spent one of the 2^ℓ coins in her wallet, *all* her spendings of these coins can be traced. However, the price for this is that the complexity of the spending and of the withdrawal protocols becomes $\mathcal{O}(\ell \cdot k)$ and $\mathcal{O}(\ell \cdot k + k^2)$ bits, respectively, and wallets take $\mathcal{O}(\ell \cdot k)$ bits of storage. All our schemes are secure in the random oracle model.

1 Introduction

Electronic cash was invented by Chaum [22, 23], and extensively studied since, e.g., [24, 26, 37, 7, 41, 20, 21]. The main idea is that, even though the same party (a bank $\mathcal{B}$) is responsible for giving out electronic coins, and for later accepting them for deposit, the withdrawal and the spending protocols are designed in such a way that it is impossible to identify when a particular coin was spent. I.e., the withdrawal protocol does not reveal any information to the bank that would later enable it to trace how a coin was spent.

As a coin is represented by data, and it is easy to duplicate data, an electronic cash scheme requires a mechanism that prevents a user from spending the same

⋆ Research performed while at IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland.

⋆⋆ Supported by NSF Career Grant CNS-0347661.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 302–321, 2005.

coin twice (double-spending). There are two scenarios. In the *on-line* scenario [23, 24, 25], the bank is on-line in each transaction to ensure that no coin is spent twice, and each merchant must consult the bank before accepting a payment. In the *off-line* [26] scenario, the merchant accepts a payment autonomously, and later submits the payment to the bank; the merchant is guaranteed that such a payment will be either honored by the bank, or will lead to the identification (and therefore punishment) of the double-spender.

In this paper, we give an off-line 2^ℓ-spendable unlinkable electronic cash scheme. Namely, our scheme allows a user to withdraw a wallet with 2^ℓ coins, such that the space required to store these coins, and the complexity of the withdrawal protocol, are proportional to ℓ, rather than to 2^ℓ. We achieve this without compromising the anonymity and unlinkability properties usually required of electronic cash schemes. This problem is well-motivated: (1) communication with the bank is a bottleneck in most electronic cash schemes and needs to be minimized; (2) it is desirable to store many electronic coins compactly, as one can imagine that they may be stored on a dedicated device such as a smartcard that cannot store too much data. This problem has also proved quite elusive: no one has offered a compact e-cash solution (even for a weaker security model) since the introduction of electronic cash in the 1980s.

In addition, a good e-cash scheme should allow one to expose double-spenders to outside third parties in an undeniable fashion. I.e., assuming a PKI, if a user $\mathcal{U}$ with public key $pk_{\mathcal{U}}$ spent a coin more times than he is allowed (in our case, spent $2^\ell + 1$ coins from a wallet containing 2^ℓ coins), then this fact can be proven to anyone in a sound fashion. This property of an e-cash scheme is satisfied by numerous schemes in the literature. Our solution has this property as well.

Finally, it may often be desirable that an e-cash scheme should allow one to trace *all* coins of a cheating user. It was known that this property can be implemented using a trusted third party (TTP) [40, 10], by requiring that: (1) in each withdrawal protocol a user gives to the bank an encryption under the TTP's public key of a serial number S which will be revealed during the spending protocol; and (2) in each spending protocol, the user submits to the merchant an encryption of the user's public key under the TTP's public key. Then, should a coin with serial number S ever be double-spent, the TTP can get involved and decrypt the serial number of all of this user's coins. But the existence of such a TTP contradicts the very definition of electronic cash: to the TTP, the user is not anonymous! Therefore, another desirable and elusive property of an electronic cash scheme was traceability *without* a TTP. Our scheme achieves this property as well.

Recently, Jarecki and Shmatikov [34] also made a step in this direction. Although their work is not explicitly about electronic cash, it can be thought of in this way. Their scheme allows to withdraw and *linkably* (linkability is actually a feature for them) but anonymously spend a coin K times; but should a user wish to spend the coin $K+1$ times, his identity gets revealed. As far as electronic cash is concerned, our solution is better for two reasons: (1) their scheme does not

achieve unlinkability; and (2) in their protocol, each time a user spends a coin he has to run a protocol whose communication complexity is proportional to K, rather than $\log K$, as we achieve. In 1989, Okamoto and Ohta [37] proposed an e-cash scheme with similar functionality, without achieving unlinkability or compact wallets.

Our work can also be viewed as improving on the recent traceable group signatures by Kiayias, Tsiounis, and Yung [35]. In their scheme, once a special piece of tracing information is released, it is possible to trace all group signatures issued by a particular group member; otherwise this member's signatures are guaranteed to remain anonymous. Normally, in a group signature setting, this piece of information *must* be released by a TTP, as there is no equivalent of a *double-spender* whose misbehavior may automatically lead to the release of the tracing information; however, if a limit is placed on how many signatures a group member may issue, then our e-cash scheme can be viewed as a *bounded* group signature scheme, where a group member can sign a message by incorporating it into the signature proof of a coin's validity. A group manager may allocate signing rights by acting as a bank allocating coins; and if any member exceeds their allocation, the special tracing information is revealed automatically, and all signatures produced by that group member may be traced. Our tracing algorithm is more efficient than that of Kiayias et al. [35]; in our scheme, signatures can be tracked by a serial number (that appears to be random until the user double-spends), while in theirs, *all* existing signatures must be tested, one-by-one, using the special tracing information provided by the TTP, to determine if a certain signer created it or not.

Our results. Let us summarize our results. We give a compact e-cash scheme with all the features described above in the random-oracle model, under the Strong RSA assumption in combination with the decisional Diffie-Hellman inversion (y-DDHI) [4, 31] and sum-free DDH [30] assumptions for groups with bilinear maps. The communication complexity of the spending and of the withdrawal protocol is $\mathcal{O}(\ell \cdot k)$ and $\mathcal{O}(\ell \cdot k + k^2)$ bits, respectively; it takes $\mathcal{O}(\ell \cdot k)$ bits to store all the coins. This scheme is presented in Section 4.2.

We also give a scheme where the withdrawal and the spending protocols have complexity only $\mathcal{O}(\ell + k)$, and it also takes only $\mathcal{O}(\ell + k)$ bits to store all the coins, based on the Strong RSA [33, 3] and the y-DDHI [31] assumptions in the random-oracle model. This less expensive scheme does not allow traceability, however. This scheme is presented in Section 4.1.

Furthermore, in the model where the bank completely trusts the merchant (this applies to, for example, a subscription service where the entity creating and verifying the coins is one and the same), we have solutions based on the same set of assumptions but *in the standard model*. Sections 4.1 and 4.2 containing our random-oracle-based schemes also explain how these security properties are obtained once the random oracle is removed.

Overview of our construction. Our schemes are based on the signature schemes with protocols due to Camenisch and Lysyanskaya [14, 15]. These schemes allow

a user to efficiently obtain a signature on committed messages from the signer. They further allow the user to convince a verifier that she possesses a signature by the signer on a committed message. Both of these protocols rely on the Pedersen commitment scheme.

To explain our result, let us describe how single-use electronic cash can be obtained with CL-signatures, drawing on a variety of previously known techniques [9, 14].

Let $G = \langle g \rangle$ be a group of prime order q where the discrete logarithm problem is hard. Suppose that a user $\mathcal{U}$ has a secret key $sk_{\mathcal{U}} \in \mathbb{Z}_q$ and a public key $pk_{\mathcal{U}} = g^{sk_{\mathcal{U}}}$. An electronic coin is a signature under the bank $\mathcal{B}$'s public key $pk_{\mathcal{B}}$ on the set of values $(sk_{\mathcal{U}}, s, t)$, where $s, t \in \mathbb{Z}_q$ are random values. The value s is the *serial number* of the coin, while t is the *value blinding* of this coin. A protocol whereby a user obtains such a signature is called the *withdrawal protocol*.

In the *spending protocol*, the user sends the merchant a Pedersen commitment C to the values $(sk_{\mathcal{U}}, s, t)$, and computes a non-interactive proof π_1 that they have been signed by the bank. The merchant verifies π_1 and then picks a random value $R \in \mathbb{Z}_q$. Finally, the user reveals the serial number s, and the value $T = sk_{\mathcal{U}} + R \cdot t \bmod q$. Let us refer to T as a *double-spending equation* for the coin. The user must also compute a proof π_2 that the values s and T correspond to commitment C. Finally, the merchant submits (s, R, T, π_1, π_2) for payment.

Note that one double-spending equation reveals nothing about $sk_{\mathcal{U}}$ because t is random, but using two double-spending equations, we can solve for $sk_{\mathcal{U}}$. So if the same serial number s is submitted for payment twice, the secret key $sk_{\mathcal{U}}$ and therefore the identity of the double-spender $pk_{\mathcal{U}} = g^{sk_{\mathcal{U}}}$ can be discovered.

Now, our goal is to adapt single-use electronic cash schemes so that a coin can be used at most 2^ℓ times. The trivial solution would be to obtain 2^ℓ coins. For our purposes, however, it is unacceptable, as 2^ℓ may be quite large (e.g., 1000) and we want each protocol to be efficient.

The idea underlying our system is that the values s and t implicitly define several (pseudorandom) serial numbers S_i and blinding values B_i, respectively. In other words, we need a pseudorandom function $F_{(\cdot)}$ such that we can set $S_i = F_s(i)$, and $B_i = F_t(i)$, $0 \leq i \leq 2^\ell - 1$. Then the user gets 2^ℓ pseudorandom serial numbers with the corresponding double-spending equations defined by (s, t). Here, the double-spending equation for coin i is $T_i = g^{sk_{\mathcal{U}}}(B_i)^R$, where R is chosen by the merchant. This leaves us with a very specific technical problem. The challenge is to find a pseudorandom function such that, given (1) a commitment to $(sk_{\mathcal{U}}, s, t)$; (2) a commitment to i; and (3) the values S_i and T_i, the user can efficiently prove that she derived the values S_i and T_i correctly from $sk_{\mathcal{U}}$, s, and t, i.e., $S_i = F_s(i)$ and $T_i = g^{sk_{\mathcal{U}}}(F_t(i))^{R_i}$ for some $0 \leq i \leq 2^\ell - 1$ and public value R_i provided by the merchant.

Recently, Dodis and Yampolsky [31] proposed the following discrete-logarithm-based pseudorandom function (PRF): $F_s(x) = g^{1/(s+x+1)}$, where $s, x \in \mathbb{Z}_q$, and g is a generator of a group G of order q in which the decisional Diffie-Hellman

inversion problem is hard.[1] (In the sequel, we denote this PRF as $F^{DY}_{(\cdot)}(\cdot)$.) Using standard methods for proving statements about discrete-logarithm representations, we obtain a zero-knowledge argument system for showing that a pair of values (S_i, T_i) is of the form $S_i = F^{DY}_s(i)$ and $T_i = g^{sk_{\mathcal{U}}}(F^{DY}_t(i))^{R_i}$ corresponding to the seeds s and t signed by bank $\mathcal{B}$ and to some index $i \in [0, 2^\ell - 1]$.

Note that if S_i and T_i are computed this way, then they are elements of G rather than of $\mathbb{Z}_q$. So this leaves us with the following protocol: to withdraw a coin, a user obtains a signature on $(sk_{\mathcal{U}}, s, t)$. During the spending protocol, the user reveals S_i and the double-spending equation $T_i = g^{sk_{\mathcal{U}}}(B_i)^{R_i}$, where $sk_{\mathcal{U}}$ is the user's secret key and $pk_{\mathcal{U}} = g^{sk_{\mathcal{U}}}$ the corresponding public key. Now, with two double-spending equations $T_1 = g^{sk_{\mathcal{U}}} B_i^{R_1}$ and $T_2 = g^{sk_{\mathcal{U}}} B_i^{R_2}$ we can infer the value $(T_1^{R_2}/T_2^{R_1})^{(R_2-R_1)^{-1}} = (pk_{\mathcal{U}}^{R_2} B_i^{R_1 R_2}/pk_{\mathcal{U}}^{R_1} B_i^{R_1 R_2})^{(R_2-R_1)^{-1}} = (pk_{\mathcal{U}}^{R_2-R_1})^{(R_2-R_1)^{-1}} = pk_{\mathcal{U}}$. This is sufficient to detect and identify double spenders. We describe this construction in more depth in Section 4.1.

However, the above scheme does not allow the bank to identify the other spendings of the coin, i.e., to generate all the serial numbers that the user can derive from s. Let us now describe how we achieve this. For the moment, let us assume that the technique described above allows us to infer $sk_{\mathcal{U}}$ rather than $pk_{\mathcal{U}}$. If this were the case, we could require that the user, as part of the withdrawal protocol, should verifiably encrypt [1, 11, 18] the value s *under her own* $pk_{\mathcal{U}}$, to form a ciphertext c. The record $(pk_{\mathcal{U}}, c)$ is stored by the bank. Now, suppose that at a future point, the user spends too many coins and thus her $sk_{\mathcal{U}}$ is discovered. From this, her $pk_{\mathcal{U}}$ can be inferred and the record $(pk_{\mathcal{U}}, c)$ can be located. Now that $sk_{\mathcal{U}}$ is known, c can be decrypted, the seed s discovered, the values S_i computed for all $0 \le i < 2^\ell$, and hence the database of transactions can be searched for records with these serial numbers.

Let us now redefine the way a user's keys are picked such that we can recover $sk_{\mathcal{U}}$ rather than $pk_{\mathcal{U}}$. Suppose that G is a group with a non-degenerate bilinear map $e : G \times G \mapsto G'$. Let $sk_{\mathcal{U}}$ be an element of $\mathbb{Z}_q$. Let $pk_{\mathcal{U}} = e(g, g^{sk_{\mathcal{U}}})$. Recently, Ateniese, Fu, Green, and Hohenberger [2] exhibited a cryptosystem that uses $pk_{\mathcal{U}}$ as a public key, such that in order to decrypt it is sufficient to know the value $g^{sk_{\mathcal{U}}}$. So, in our scheme, the user $\mathcal{U}$ would encrypt s under $pk_{\mathcal{U}}$ using the cryptosystem due to Ateniese et al. From the double-spending equations, the same way as before, the bank infers the value $g^{sk_{\mathcal{U}}}$. This value now allows the bank to decrypt s.

This is almost the solution, except for the following subtlety: if G has a bilinear map, then the decisional Diffie-Hellman problem is easy, and so the Dodis-Yampolsky construction is not a PRF in this setting! Instead, we must assume sum-free decisional Diffie-Hellman [30], and slightly change the construction. This is why the variant of our scheme that allows to trace coins is a factor

[1] Another PRF suitable for our purposes is the one due to Naor and Reingold [36]. It is based on the DDH problem, but makes our subsequent schemes less time and space efficient.

of ℓ more expensive than the one that does not. The details of this construction are given in Section 4.2.

One of the main remaining problems for electronic cash which this paper does not address is that of efficiently allowing for multiple denominations in a non-trivial way; i.e., without executing the spending protocol a number of times.

2 Definition of Security

Notation: if P is a protocol between A and B, then $P(A(x), B(y))$ denotes that A's input is x and B's is y.

Our electronic cash scenario consists of the three usual players: the user, the bank, and the merchant; together with the algorithms: $\mathsf{BKeygen}$, $\mathsf{UKeygen}$, $\mathsf{Withdraw}$, Spend, $\mathsf{Deposit}$, $\mathsf{Identify}$, $\mathsf{VerifyGuilt}$, Trace, $\mathsf{VerifyOwnership}$. Let us give some input-output specifications for these protocols, as well as some informal intuition for what they do.

- The $\mathsf{BKeygen}(1^k, params)$ algorithm is a key generation algorithm for the bank $\mathcal{B}$. It takes as input the security parameter 1^k and, if the scheme is in the common parameters model, it also takes as input these parameters $params$. This algorithm outputs the key pair $(pk_{\mathcal{B}}, sk_{\mathcal{B}})$. (Assume that $sk_{\mathcal{B}}$ contains the $params$, so we do not have to give $params$ explicitly to the bank again.)
  Similarly, $\mathsf{UKeygen}(1^k, params)$ is a key generation algorithm for the user $\mathcal{U}$, which outputs $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$. Since merchants are a subset of users, they may use this algorithm to obtain keys as well. (Assume that sk contains the $params$, so we do not have to give $params$ explicitly to the user again.)
- In the $\mathsf{Withdraw}(\mathcal{U}(pk_{\mathcal{B}}, sk_{\mathcal{U}}, n), \mathcal{B}(pk_{\mathcal{U}}, sk_{\mathcal{B}}, n))$ protocol, the user $\mathcal{U}$ withdraws a *wallet* W of n coins from the bank $\mathcal{B}$. The user's output is the wallet W, or an error message. $\mathcal{B}$'s output is some information T_W which will allow the bank to trace the user should this user double-spend some coin, or an error message. The bank maintains a database D for this trace information, to which it enters the record $(pk_{\mathcal{U}}, T_W)$.
- In a $\mathsf{Spend}(\mathcal{U}(W, pk_{\mathcal{M}}), \mathcal{M}(sk_{\mathcal{M}}, pk_{\mathcal{B}}, n))$ protocol, a user $\mathcal{U}$ gives one of the coins from his wallet W to the merchant $\mathcal{M}$. Here, the merchant obtains a serial number S of the coin, and a proof π of validity of the coin. The user's output is an updated wallet W'.
- In a $\mathsf{Deposit}(\mathcal{M}(sk_{\mathcal{M}}, S, \pi, pk_{\mathcal{B}}), \mathcal{B}(pk_{\mathcal{M}}, sk_{\mathcal{B}}))$ protocol, a merchant $\mathcal{M}$ deposits a coin (S, π) into its account held by the bank $\mathcal{B}$. Whenever an honest $\mathcal{M}$ obtained (S, π) by running the Spend protocol with any (honest or otherwise) user, there is a guarantee that this coin will be accepted by the bank. $\mathcal{B}$ adds (S, π) to to its list L of spent coins. The merchant's output is nothing or an error message.
- The $\mathsf{Identify}(params, S, \pi_1, \pi_2)$ algorithm allows to identify double-spenders using a serial number S and two proofs of validity of this coin, π_1 and π_2, possibly submitted by malicious merchants. This algorithm outputs a public key pk_U and a proof Π_G. If the merchants who had submitted π_1 and π_2 are

not malicious, then Π_G is evidence that $pk_{\mathcal{U}}$ is the registered public key of a user that double-spent coin S.

- The $\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$ algorithm allows to publicly verify proof Π_G that the user with public key $pk_{\mathcal{U}}$ is guilty of double-spending coin S.
- The $\mathsf{Trace}(params, S, pk_{\mathcal{U}}, \Pi_G, D, n)$ algorithm, given a public key $pk_{\mathcal{U}}$ of a double-spender, a proof Π_G of his guilt in double-spending coin S, the database D, and a wallet size n, computes the serial numbers $S_1, \ldots, S_m$ of all of the coins issued to $\mathcal{U}$ along with proofs $\Pi_1, \ldots, \Pi_m$ of $pk_{\mathcal{U}}$'s ownership. If $\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$ does not accept (i.e., $pk_{\mathcal{U}}$ is honest), this algorithm does nothing.
- The $\mathsf{VerifyOwnership}(params, S, \Pi, pk_{\mathcal{U}}, n)$ algorithm allows to publicly verify the proof Π that a coin with serial number S belongs to a double-spender with public key $pk_{\mathcal{U}}$.

We will now informally define the security properties. The more elaborate formal definitions are given in the full version of this paper [13].

Correctness. If an honest user runs Withdraw with an honest bank, then neither will output an error message; if an honest user runs Spend with an honest merchant, then the merchant accepts the coin.

Balance. From the bank's point of view, what matters is that no collection of users and merchants can ever spend more coins than they withdrew. We require that there is a knowledge extractor $\mathcal{E}$ that executes u Withdraw protocols with all adversarial users and extracts un serial numbers $S_1, \ldots, S_{un}$. We require that for every adversary, the probability that an honest bank will accept (S, π) as the result of the Deposit protocol, where $S \neq S_i \ \forall 1 \leq i \leq un$, is negligible. If $S_1, \ldots, S_n$ is a set of serial numbers output by $\mathcal{E}$ when running Withdraw with public key $pk_{\mathcal{U}}$, we say that coins $S_1, \ldots, S_n$ *belong* to the user $\mathcal{U}$ with $pk_{\mathcal{U}}$.

Identification of double-spenders. Suppose $\mathcal{B}$ is honest. Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ are honest merchants who ran the Spend protocol with the adversary, such that $\mathcal{M}_1$'s output is (S, π_1) and $\mathcal{M}_2$'s output is (S, π_2). This property guarantees that, with high probability, $\mathsf{Identify}(params, S, \pi_1, \pi_2)$ outputs a key $pk_{\mathcal{U}}$ and proof Π_G such that $\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$ accepts.

Tracing of double-spenders. Given that a user $\mathcal{U}$ is shown guilty of double-spending coin S by a proof Π_G such that VerifyGuilt accepts, this property guarantees that $\mathsf{Trace}(params, S, pk_{\mathcal{U}}, \Pi_G, D, n)$ will output the serial numbers $S_1, \ldots, S_m$ of all coins that belong to $\mathcal{U}$ along with proofs of ownership $\Pi_1, \ldots, \Pi_m$ such that for all i, with high probability, $\mathsf{VerifyOwnership}(params, S_i, \Pi_i, pk_{\mathcal{U}}, n)$ also accepts.

Anonymity of users. From the privacy point of view, what matters to users is that the bank, even when cooperating with any collection of malicious users and merchants, cannot learn anything about a user's spendings other than what is available from side information from the environment. In order to capture this property more formally, we introduce a simulator $\mathcal{S}$. $\mathcal{S}$ has some side information

not normally available to players. E.g., if in the common parameters model, $\mathcal{S}$ generated these parameters; in the random-oracle model, $\mathcal{S}$ is in control of the random oracle; in the public-key registration model $\mathcal{S}$ may hold additional information about the bank's keys, etc. We require that $\mathcal{S}$ can create simulated coins without access to any wallets, such that a simulated coin is indistinguishable from a valid one. More precisely, $\mathcal{S}$ executes the user's side of the Spend protocol without access to the user's secret or public key, or his wallet W.

Exculpability. Suppose that we have an adversary that participates any number of times in the Withdraw protocol with the honest user with public key $pk_{\mathcal{U}}$, and subsequently to that, in any number of legal Spend protocols with the same user. I.e., if the user withdrew u wallets of n coins each, then this user can participate in at most un Spend protocols. The adversary then outputs a coin serial number S and a purported proof Π that the user with public key $pk_{\mathcal{U}}$ is a double-spender and owns coin S. The *weak* exculpability property postulates that, for all adversaries, the probability $\mathsf{VerifyOwnership}(params,\ S,\ pk_{\mathcal{U}}, \Pi, n)$ accepts is negligible.

Furthermore, the adversary may continue to engage the user $\mathcal{U}$ in Spend protocols even if it means $\mathcal{U}$ must double-spend some coins of her choosing (in which case the state of her wallet is reset). The adversary then outputs (S, Π). The *strong* exculpability property postulates that, for all adversaries, when S is a coin serial number *not* belonging to $\mathcal{U}$, the weak exculpability property holds, and when S is a coin serial number *not* double-spent by user $\mathcal{U}$ with public key $pk_{\mathcal{U}}$, the probability that $\mathsf{VerifyGuilt}(params, S, \Pi, pk_{\mathcal{U}}, n)$ accepts is negligible.

This ends the informal description of our security definition; the descriptions in the full version of this paper [13] are more precise, but this intuition should be sufficient for understanding our subsequent security guarantees.

Strengthening the definition: the UC framework. Even though our definition of security is not in the UC framework, note that our definition would imply UC-security whenever the extractor $\mathcal{E}$ and simulator $\mathcal{S}$ are constructed appropriately. In a nutshell, an ideal electronic cash functionality would allow an honest user to withdraw and spend n coins. In this case, if the merchant and bank are controlled by the malicious environment, the simulator $\mathcal{S}$ defined above creates the merchant's and bank's view of the Spend protocol. At the same time, the balance property guarantees that the bank gets the same protection in the real world as it does in the ideal world, and the exculpability property ensures that an honest user cannot get framed in the real world, just as he cannot get framed in the ideal world.

3 Preliminaries

Our e-cash systems use a variety of known protocols as building blocks, which we now briefly review. Many of these protocols can be shown secure under several

different complexity assumptions, a flexibility that will extend to our e-cash systems. The notation $G = \langle g \rangle$ means that g generates the group G.

3.1 Complexity Assumptions

The security of our e-cash systems is based on the following assumptions:

Strong RSA Assumption [3, 33]: Given an RSA modulus n and a random element $g \in \mathbb{Z}_n^*$, it is hard to compute $h \in \mathbb{Z}_n^*$ and integer $e > 1$ such that $h^e \equiv g \mod n$. The modulus n is of a special form pq, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes.

y-Decisional Diffie-Hellman Inversion Assumption (y-DDHI) [4, 31]: Given a random generator $g \in G$, where G has prime order q, the values $(g, g^x, \ldots, g^{(x^y)})$ for a random $x \in \mathbb{Z}_q$, and a value $R \in G$, it is hard to decide if $R = g^{1/x}$ or not.[2]

Sum-Free Decisional Diffie-Hellman Assumption (SF-DDH) [30]: Suppose that $g \in G$ is a random generator of order q. Let L be any polynomial function of $|q|$. Let $O_{\boldsymbol{a}}(\cdot)$ be an oracle that, on input a subset $I \subseteq \{1, \ldots, L\}$, outputs the value $g_1^{\beta_I}$ where $\beta_I = \prod_{i \in I} a_i$ for some $\boldsymbol{a} = (a_1, \ldots, a_L) \in \mathbb{Z}_q^L$. Further, let R be a predicate such that $R(J, I_1, \ldots, I_t) = 1$ if and only if $J \subseteq \{1, \ldots, L\}$ is DDH-independent from the I_i's; that is, when $v(I_i)$ is the L-length vector with a one in position j if and only if $j \in I_i$ and zero otherwise, then there are no three sets I_a, I_b, I_c such that $v(J) + v(I_a) = v(I_b) + v(I_c)$ (where addition is bitwise over the integers). Then, for all probabilistic polynomial time adversaries $\mathcal{A}^{(\cdot)}$,

$$Pr[\boldsymbol{a} = (a_1, \ldots, a_L) \leftarrow \mathbb{Z}_q^L; (J, \alpha) \leftarrow \mathcal{A}^{O_{\boldsymbol{a}}}(1^{|q|}); y_0 = g^{\prod_{i \in J} a_i}; y_1 \leftarrow G;$$
$$b \leftarrow \{0,1\}; b' \leftarrow \mathcal{A}^{O_{\boldsymbol{a}}}(1^{|q|}, y_b, \alpha) \ : \ b = b' \wedge R(J, Q) = 1] = negl(|q|),$$

where Q is the set of queries that $\mathcal{A}$ made to $O_{\boldsymbol{a}}(\cdot)$.

3.2 Bilinear Maps

Let Bilinear_Setup be an algorithm that, on input the security parameter 1^k, outputs $\gamma = (q, g_1, h_1, G_1, g_2, h_2, G_2, e)$, where e is a non-degenerate efficiently computable bilinear map from $G_1 = \langle g_1 \rangle = \langle h_1 \rangle$ to $G_2 = \langle g_2 \rangle = \langle h_2 \rangle$, both groups of prime order $q = \Theta(2^k)$. Let $e(g_1, g_1) = g_2$ and $e(h_1, h_1) = h_2$. We assume that each group element has a unique binary representation. More formally, $e : G_1 \times G_1 \rightarrow G_2$ is a function that is: (bilinear) for all $g_1, h_1 \in G_1$, for all $a, b \in \mathbb{Z}_q$, $e(g_1^a, h_1^b) = e(g_1, h_1)^{ab}$; (non-degenerate) if g_1 is a generator of G_1, then $e(g_1, g_1)$ generates G_2; and (efficient) computing $e(\cdot, \cdot)$ is efficient for all $g_1, h_1 \in G_1$.

[2] Others [4, 31] have used a stronger *bilinear* version of the y-DDHI assumption, where, given the same input in $\langle g \rangle^{y+1}$ it is hard to distinguish $e(g, g)^{1/x}$ from a random R in $\langle e(g, g) \rangle$.

3.3 Known Discrete-Logarithm-Based, Zero-Knowledge Proofs

In the common parameters model, we use several previously known results for proving statements about discrete logarithms, such as (1) proof of knowledge of a discrete logarithm modulo a prime [39] or a composite [33, 29], (2) proof of knowledge of equality of representation modulo two (possibly different) prime [27] or composite [17] moduli, (3) proof that a commitment opens to the product of two other committed values [16, 20, 8], (4) proof that a committed value lies in a given integer interval [21, 16, 16, 6], and also (5) proof of the disjunction or conjunction of any two of the previous [28]. These protocols modulo a composite are secure under the strong RSA assumption and modulo a prime under the discrete logarithm assumption.

When refering to the proofs above, we will follow the notation introduced by Camenisch and Stadler [19] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(\alpha, \beta, \delta) : y = g^{\alpha} h^{\beta} \wedge \tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\delta} \wedge (u \leq \alpha \leq v)\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers* α*,* β*, and* δ *such that* $y = g^{\alpha} h^{\beta}$ *and* $\tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\delta}$ *holds, where* $u \leq \alpha \leq v$*,*" where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote quantities of which knowledge is being proven, while all other values are known to the verifier. We apply the Fiat-Shamir heuristic [32] to turn such proofs of knowledge into signatures on some message m; denoted as, e.g., $SPK\{(\alpha) : y = g^{\alpha}\}(m)$.

3.4 Pseudorandom Functions

A useful building block of our e-cash systems is the pseudorandom functions recently proposed by Dodis and Yampolsky [31], which they expand to verifiable random functions using bilinear maps. Their construction is:

For every n, a function $f \in F_n$ is defined by the tuple (G, q, g, s), where G is group of order q, q is an n-bit prime, g is a generator of G, and s is a seed in $\mathbb{Z}_q$. For any input $x \in \mathbb{Z}_q$ (except for $x = -1 \bmod q$), the function $f_{P,q,g,s}(\cdot)$, which we simply denote as $f_s^{DY}(\cdot)$, is defined as $f_s^{DY}(x) = g^{1/(x+s+1)}$.

This construction is secure under the y-DDHI assumption. As mentioned in the introduction, we could instead substitute in the Naor-Reingold PRF [36], and replace the y-DDHI assumption with the more standard DDH assumption, at the cost of enlarging our wallets from $\mathcal{O}(\ell + k)$ bits to $\mathcal{O}(\ell \cdot k)$ bits.

3.5 CL Signatures

Recall the Pedersen commitment scheme [38], in which the public parameters are a group G of prime order q, and generators $(g_0, \ldots, g_m)$. In order to commit to the values $(v_1, \ldots, v_m) \in {\mathbb{Z}_q}^m$, pick a random $r \in \mathbb{Z}_q$ and set $C = \text{PedCom}(v_1, \ldots, v_m; r) = g_0^r \prod_{i=1}^m g_i^{v_i}$.

Camenisch and Lysyanskaya [14] came up with a secure signature scheme with two protocols: (1) An efficient protocol between a user and a signer with

keys (pk_S, sk_S). The common input consists of pk_S and C, a Pedersen commitment. The user's secret input is the set of values $(v_1, \ldots, v_\ell, r)$ such that $C = \text{PedCom}(v_1, \ldots, v_\ell; r)$. As a result of the protocol, the user obtains a signature $\sigma_{pk_S}(v_1, \ldots, v_\ell)$ on his committed values, while the signer does not learn anything about them. The signature has size $\mathcal{O}(\ell \log q)$. (2) An efficient proof of knowledge of a signature protocol between a user and a verifier. The common inputs are pk_S and a commitment C. The user's private inputs are the values $(v_1, \ldots, v_\ell, r)$, and $\sigma_{pk_S}(v_1, \ldots, v_\ell)$ such that $C = \text{PedCom}(v_1, \ldots, v_\ell; r)$. These signatures are secure under the strong RSA assumption. For the purposes of this exposition, it does not matter *how* CL signatures actually work, all that matters are the facts stated above.

Our subsequent e-cash systems will require the strong RSA assumption independently of the CL signatures. By making additional assumptions based on bilinear maps, we can use alternative schemes by Camenisch and Lysyanskaya [15] and Boneh, Boyen and Shacham [5], yielding shorter signatures in practice.

3.6 Verifiable, Bilinear El Gamal Encryption

In Section 4.2, we apply a technique by Camenisch and Damgård [11] for turning any semantically-secure encryption scheme into a verifiable encryption scheme. A verifiable encryption scheme is a two-party protocol between a prover and encryptor $\mathcal{P}$ and a verifier and receiver $\mathcal{V}$. Roughly, their common inputs are a public encryption key pk and a commitment A. As a result of the protocol, $\mathcal{V}$ either rejects or obtains the encryption c of the opening of A. The protocol ensures that $\mathcal{V}$ accepts an incorrect encryption only with negligible probability and that $\mathcal{V}$ learns nothing meaningful about the opening of A. Together with the corresponding secret key sk, transcript c contains enough information to recover the opening of A efficiently. We hide some details here and refer to Camenisch and Damgård [11] for the full discussion.

In particular, we apply the verifiable encryption techniques above to a bilinear variant of El Gamal encryption due to Ateniese, Fu, Green, and Hohenberger [2]. Assume we run $\mathsf{Bilinear\_Setup}$ on 1^k to obtain $\zeta = (q, \mathsf{g}_3, \mathcal{G}_3, \mathsf{g}_2, \mathcal{G}_2, e)$, where we have bilinear map $e : \mathcal{G}_3 \times \mathcal{G}_3 \to \mathcal{G}_2$. Let (G, E, D) denote the standard key generation, encryption, and decryption algorithms. On input $(1^k, \zeta)$, the key generation algorithm G outputs a key pair $(pk, sk) = (e(\mathsf{g}_3, \mathsf{g}_3^u), \mathsf{g}_3^u) = (\mathsf{g}_2^u, \mathsf{g}_3^u)$ for a random $u \in \mathbb{Z}_q$. To encrypt a message $m \in \mathcal{G}_2$ under pk, select a random $k \in \mathbb{Z}_q$ and output the ciphertext $c = (\mathsf{g}_3^k, pk^k m) = (\mathsf{g}_3^k, \mathsf{g}_2^{uk} m)$. Then, to decrypt $c = (c_1, c_2)$, simply compute $c_2 / e(c_1, sk)$. Ateniese et al. [2] show that this encryption scheme is semantically-secure under the decisional bilinear DH assumption.

4 Two Compact E-Cash Systems

We present two compact e-cash systems. In System One, an honest bank can quickly detect double-spending, identify the perpetrator, and prove his guilt to a third party from two coin deposits with the same serial number. This system

allows a wallet of 2^ℓ coins to be stored in $\mathcal{O}(\ell+k)$ bits. In System Two, the bank can do everything that it could before, and in addition, the bank can compute the serial numbers for all the coins that belong to the perpetrator along with proofs of their ownership. Here, our wallets of 2^ℓ coins require $\mathcal{O}(\ell \cdot k)$ bits, which is still remarkably small. If a user does not double-spend, her coins are unlinkable. There is no trusted party.

Global parameters for both systems. Let 1^k be the security parameter and let ℓ be any value in $\mathcal{O}(\log k)$. Our subsequent schemes work most efficiently by having three different algebraic groups:

- $\mathcal{G}_1 = \langle g_1 \rangle$, where n is a special RSA modulus of $2k$ bits, g_1 is a quadratic residue modulo n, and $h_1 \in \mathcal{G}_1$.
- $\mathcal{G}_2 = \langle \mathsf{g}_2 \rangle$, where g_2 is an element of prime order $q = \Theta(2^k)$, and $\mathsf{h}_2 \in \mathcal{G}_2$.
- $\mathcal{G}_3 = \langle \mathsf{g}_3 \rangle$, where g_3 is an element of the same prime order as $\mathcal{G}_2$, and there exists a bilinear mapping $e : \mathcal{G}_3 \times \mathcal{G}_3 \rightarrow \mathcal{G}_2$.

Our first scheme will not require $\mathcal{G}_3$. Assume that, on input 1^k, each system is initialized with the necessary common parameters, denoted ζ. We also define the multiple Pedersen commitment as $\mathrm{PedCom}(x_1, \ldots, x_n; r) = h^r \Pi_{i=1}^n g_i^{x_i}$. Sometimes for simplicity we do not explicitly include the randomess r in the input to the commitment. The values $h, \{g_i\}$ are assumed to be publicly known elements of the appropriate group.

4.1 System One: Wallets of Size $\mathcal{O}(\ell + k)$ with Public Key Recovery

Our first system supports the basic algorithms ($\mathsf{BKeygen}, \mathsf{UKeygen}, \mathsf{Withdraw}$, Spend, $\mathsf{Deposit}$, $\mathsf{Identify}$, $\mathsf{VerifyGuilt}$). In this scheme, a wallet of size $\mathcal{O}(\ell + k)$ is sufficient to hold 2^ℓ coins. In the $\mathsf{Identify}$ algorithm, the bank can recover the identity of a double-spender $pk_{\mathcal{U}}$ from two deposits with the same coin serial number S. Using $\mathsf{VerifyGuilt}$, the bank can prove that $pk_{\mathcal{U}}$ double-spent coin S to a third party; while all honest users are guaranteed strong exculpability.

The parties set up their keys as follows. In $\mathsf{BKeygen}(1^k, \zeta)$, the bank $\mathcal{B}$ generates a CL signature key pair $(pk_{\mathcal{B}}, sk_{\mathcal{B}})$ for message space M such that $\mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q \subseteq M$. In $\mathsf{UKeygen}(1^k, \zeta)$, each user $\mathcal{U}$ generates a unique key pair $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) = (\mathsf{g}_2^u, u)$ for a random $u \in \mathbb{Z}_q$. Recall that merchants are a subset of users.

$\mathsf{Withdraw}(\mathcal{U}(pk_{\mathcal{B}}, sk_{\mathcal{U}}, 2^\ell), \mathcal{B}(pk_{\mathcal{U}}, sk_{\mathcal{B}}, 2^\ell))$**:** A user $\mathcal{U}$ interacts with the bank $\mathcal{B}$ as follows:

1. $\mathcal{U}$ identifies himself to the bank $\mathcal{B}$ by proving knowledge of $sk_{\mathcal{U}}$.
2. In this step, the user and bank contribute randomness to the wallet secret s; the user also selects a wallet secret t. This is done as follows: $\mathcal{U}$ selects random values $s', t \in \mathbb{Z}_q$ and sends a commitment $A' = \mathrm{PedCom}(u, s', t; r)$ to $\mathcal{B}$. $\mathcal{B}$ sends a random $r' \in \mathbb{Z}_q$. Then $\mathcal{U}$ sets $s = s' + r'$. $\mathcal{U}$ and $\mathcal{B}$ locally compute $A = g_2^{r'} A' = \mathrm{PedCom}(u, s' + r', t; r) = \mathrm{PedCom}(u, s, t; r)$.

3. $\mathcal{U}$ and $\mathcal{B}$ run the CL protocol for obtaining $\mathcal{B}$'s signature on committed values contained in commitment A. As a result, $\mathcal{U}$ obtains $\sigma_{\mathcal{B}}(u, s, t)$.
4. $\mathcal{U}$ saves the wallet $W = (sk_{\mathcal{U}}, s, t, \sigma_B(u, s, t), J)$, where s, t are the wallet secrets, $\sigma_{\mathcal{B}}(u, s, t)$ is the bank's signature, and J is an ℓ-bit coin counter initialized to zero.
5. $\mathcal{B}$ records a debit of 2^ℓ coins for account $pk_{\mathcal{U}}$.

$\mathsf{Spend}(\mathcal{U}(W, pk_{\mathcal{M}}), \mathcal{M}(sk_{\mathcal{M}}, pk_{\mathcal{B}}, 2^\ell))$**:** $\mathcal{U}$ anonymously transfers a coin to $\mathcal{M}$ as follows. (An optimized version appears in the full version of this paper [13].)

1. $\mathcal{M}$ (optionally) sends a string $info \in \{0,1\}^*$ containing transaction information to $\mathcal{U}$ and authenticates himself by proving knowledge of $sk_{\mathcal{M}}$.
2. $\mathcal{M}$ chooses a random $R \in \mathbb{Z}_q^*$ and sends R to $\mathcal{U}$. This is for the double-spending equation (see Section 1).
3. $\mathcal{U}$ sends to $\mathcal{M}$ the serial number of the coin $S = F_s^{DY}(J)$, and security tag $T = pk_{\mathcal{U}} F_t^{DY}(J)^R$. Now $\mathcal{U}$ must prove their validity, i.e., that S and T correspond to wallet secrets (u, s, t) signed by $\mathcal{B}$. This is done as follows:
   (a) Let $A = \mathrm{PedCom}(J)$; prove that A is a commitment to an integer in the range $[0 \ldots 2^\ell - 1]$.
   (b) Let $B = \mathrm{PedCom}(u)$, $C = \mathrm{PedCom}(s)$, $D = \mathrm{PedCom}(t)$; prove knowledge of a CL signature from $\mathcal{B}$ on the openings of B, C and D in that order,
   (c) Prove $S = F_s^{DY}(J) = \mathsf{g}_2^{1/(J+s+1)}$ and $T = pk_{\mathcal{U}} F_t^{DY}(J)^R = \mathsf{g}_2^{u+R/(J+t+1)}$. More formally, this proof is the following proof of knowledge:

$$PK\{(\alpha, \beta, \delta, \gamma_1, \ldots, \gamma_3) : g_1 = (AC)^\alpha {h_1}^{\gamma_1} \wedge S = \mathsf{g}_2^\alpha \wedge$$
$$g_1 = (AD)^\beta {h_1}^{\gamma_2} \ \wedge \ B = {g_1}^\delta {h_1}^{\gamma_3} \ \wedge \ T = \mathsf{g}_2^\delta (\mathsf{g}_2^R)^\beta\}$$

   Use the Fiat-Shamir heuristic to turn all the proofs above into one signature of knowledge on the values $(S, T, A, B, C, D, g_1, h_1, n, \mathsf{g}_2, pk_{\mathcal{M}}, R, info)$. Call the resulting signature Φ.
4. If Φ verifies, $\mathcal{M}$ accepts the coin (S, π), where $\pi = (R, T, \Phi)$, and uses this information at deposit time.
5. $\mathcal{U}$ updates his counter $J = J + 1$. When $J > 2^\ell - 1$, the wallet is empty.

$\mathsf{Deposit}(\mathcal{M}(sk_{\mathcal{M}}, S, \pi, pk_{\mathcal{B}}), \mathcal{B}(pk_{\mathcal{M}}, sk_{\mathcal{B}}))$**:** A merchant $\mathcal{M}$ sends to bank $\mathcal{B}$ a coin $(S, \pi = (R, T, \Phi))$. If Φ verifies and R is fresh (i.e., the pair (S, R) is not already in the list L of spent coins), then $\mathcal{B}$ accepts the coin for deposit, adds (S, π) to the list L of spent coins, and credits $pk_{\mathcal{M}}$'s account; otherwise, $\mathcal{B}$ sends $\mathcal{M}$ an error message.

Note that in this deposit protocol, $\mathcal{M}$ must convince $\mathcal{B}$ that it behaved honestly in accepting some coin (S, π). As a result, our construction requires the Fiat-Shamir heuristic for turning a proof of knowledge into a signature. If $\mathcal{M}$ and $\mathcal{B}$ were the same entity, and the $\mathsf{Withdraw}$ and Spend protocols were interactive, then the bank $\mathcal{B}$ would not need to verify the validity of the coin that the

merchant wishes to deposit, and as a result, we could dispense with the Fiat-Shamir heuristic and thus achieve balance and anonymity in the plain model (i.e., not just in the random oracle model).

$\mathsf{Identify}(\zeta, S, \pi_1, \pi_2)$**:** Suppose $(R_1, T_1) \in \pi_1$ and $(R_2, T_2) \in \pi_2$ are two entries in the bank's database L of spent coins for serial number S. Then output $\Pi_G = (\pi_1, \pi_2)$ and $pk = \left(T_2^{R_1}/T_1^{R_2}\right)^{(R_1-R_2)^{-1}}$.

Let us explain why this produces the public key $pk_{\mathcal{U}}$ of the double-spender. Suppose coin S belonged to some user with $pk_{\mathcal{U}} = \mathsf{g}_2^u$, then each T_i is of the form $\mathsf{g}_2^{u+R_i\alpha}$ for the same values u and α. (Either this is true or an adversary has been successful in forging a coin, which we subsequently show happens with only negligible probability.) As the bank only accepts coins with fresh values of R (i.e., $R_1 \neq R_2$), it allows to compute:

$$\left(\frac{T_2^{R_1}}{T_1^{R_2}}\right)^{(R_1-R_2)^{-1}} = \left(\frac{\mathsf{g}_2^{uR_1+R_1R_2\alpha}}{\mathsf{g}_2^{uR_2+R_1R_2\alpha}}\right)^{(R_1-R_2)^{-1}} = \mathsf{g}_2^{\frac{u(R_1-R_2)}{(R_1-R_2)}} = \mathsf{g}_2^u = pk_{\mathcal{U}}.$$

$\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$: Parse Π_G as (π_1, π_2) and each π_i as (R_i, T_i, Φ_i). Run $\mathsf{Identify}(params, S, \pi_1, \pi_2)$ and compare the first part of its output to the public key $pk_{\mathcal{U}}$ given as input. Check that the values match. Next, verify each Φ_i with respect to (S, R_i, T_i). If all checks pass, accept; otherwise, reject.

Efficiency Discussion of System One. The dominant computational cost in these protocols are the single and multi base exponentations. In a good implementation, a multi-base exponentation is essentially as fast as an ordinary exponentation. While we do not provide the full details of the Withdraw protocol, it can easily be derived from the known protocols to obtain a CL-signature on a committed signature [14, 12]. Depending on how the proof of knowledge protocol is implemented, Withdraw requires only three moves of communication.

The details of (an optimized version of) the Spend protocol are given in the full version [13]. One can verify that a user must compute seven multi-base exponentiations to build the commitments and eleven more for the proof. The merchant and bank need to do eleven multi-base exponentiations to check that the coin is valid. The protocols require two rounds of communication between the user and the merchant and one round between the bank and the merchant.

Theorem 1. *System One supports the algorithms* (BKeygen, UKeygen, Withdraw, Spend, Deposit, Identify) *and guarantees balance, identification of double-spenders, anonymity of users, and strong exculpability under the Strong RSA and y-DDHI assumptions in the random oracle model.*

Proof of Theorem 1 appears in the full version of this paper [13].

4.2 System Two: Wallets of Size $\mathcal{O}(\ell \cdot k)$ with Traceable Coins

We now extend System One to allow coin tracing. Suppose for the moment that the Identify algorithm recovered $sk_{\mathcal{U}}$ rather than $pk_{\mathcal{U}}$ for a double-spender. We

change the withdrawal protocol so that the user $\mathcal{U}$ must also provide the bank $\mathcal{B}$ with a verifiable encryption of her wallet secret s (used to generate the coin serial numbers) under *their own public key* $pk_{\mathcal{U}}$. This way, if $\mathcal{U}$ double-spends, $\mathcal{B}$ can compute $sk_{\mathcal{U}}$, recover her secret s, and output the serial numbers $S_i = f_s^{DY}(i)$, for $i = 0$ to $2^\ell - 1$, of all coins belonging to $\mathcal{U}$. (Observe that recovering $sk_{\mathcal{U}}$ in the above scheme allows the bank to trace *all* coins from *all* wallets for $\mathcal{U}$ and not just from the wallet from which the coin was double-spent!) If $\mathcal{U}$ does not double-spend a coin, however, her anonymity is computationally guaranteed.

The verifiable encryption techniques of Camenisch and Damgård, as described in Section 3.6, can be applied to any semantically-secure encryption scheme for our withdrawal protocol. Thus, all we need is a coin construction which allows one to recover a double-spender's $sk_{\mathcal{U}}$. Luckily, the bilinear El Gamal scheme recently proposed by Ateniese et al. (see Section 3.6) allows for public keys of the form $pk_{\mathcal{U}} = e(\mathsf{g}_3, \mathsf{g}_3^u) = \mathsf{g}_2^u$, for $u \in \mathbb{Z}_q$, where knowing $sk_{\mathcal{U}} = \mathsf{g}_3^u$ is sufficient for decryption, given the mapping $e : \mathcal{G}_3 \times \mathcal{G}_3 \to \mathcal{G}_2$. By setting our tag $T_i = \mathsf{g}_3^u (f_t^{DY}(i))^R$ in $\mathcal{G}_3$ (where R is a random value chosen by the merchant), we can now recover $sk_{\mathcal{U}} = \mathsf{g}_3^u$ from a coin spent twice.

One complication with setting T in $\mathcal{G}_3$ is that the Dodis-Yampolsky [31] construction is no longer a PRF when DDH is easy, as it is in $\mathcal{G}_3$. This breaks the anonymity of our coins. Thus, we need a new PRF.

PRF based on the Sum-Free DDH Assumption. Dodis [30] gave a definition of a *sum-free* encoding, and proved that if V is any sum-free encoding, and $\langle \mathsf{g}_3 \rangle$ is a group of order q, then $f_{(\cdot)}^V$, defined as follows, is a PRF: the seed for this PRF consists of values $t_i \in \mathbb{Z}_q$, for $0 \leq i \leq 3\ell$; let $\boldsymbol{t} = (t_0, \ldots, t_{3\ell})$; the function $f_{\boldsymbol{t}}^V$ is defined as $f_{\boldsymbol{t}}^V(x) = \mathsf{g}_3^{t_0 \prod_{V(x)_i = 1} t_i}$. This holds under the sum-free DDH assumption (also introduced by Dodis [30]); note that it seems reasonable to make such an assumption even of groups where DDH is easy.

For our purposes, we need the encoding V to have nice algebraic properties. We define an encoding $V : \{0,1\}^\ell \mapsto \{0,1\}^{3\ell}$ as $V(x) = x \circ x^2$, where $\circ$ denotes concatenation, and multiplication is over the integers. In the full version of this paper, we recall the sum-free definition and prove that this encoding is sum-free.

Our second system supports all algorithms mentioned in Section 2: (BKeygen, UKeygen, Withdraw, Spend, Deposit, Identify, VerifyGuilt, Trace, VerifyOwnership). We assume a standard signature scheme $(SG, Sign, SVf)$. Then, $\mathsf{UKeygen}(1^k, \zeta)$ runs $SG(1^k, \zeta) \to (vk_{\mathcal{U}}, ssk_{\mathcal{U}})$ and the bilinear El Gamal key generation algorithm $G(1^k, \zeta) \to (ek_{\mathcal{U}}, dk_{\mathcal{U}}) = (e(\mathsf{g}_3, \mathsf{g}_3^u), \mathsf{g}_3^u) = (\mathsf{g}_2^u, \mathsf{g}_3^u)$, and outputs $pk_{\mathcal{U}} = (ek_{\mathcal{U}}, vk_{\mathcal{U}})$ and $sk_{\mathcal{U}} = (dk_{\mathcal{U}}, ssk_{\mathcal{U}})$. The bank's keys are as before.

$\mathsf{Withdraw}(\mathcal{U}(pk_{\mathcal{B}}, sk_{\mathcal{U}}, 2^\ell), \mathcal{B}(pk_{\mathcal{U}}, sk_{\mathcal{B}}, 2^\ell))$**:** A user $\mathcal{U}$ interacts with the bank $\mathcal{B}$ as follows:

1. $\mathcal{U}$ identifies himself to the bank $\mathcal{B}$ by proving knowledge of $sk_{\mathcal{U}} = (dk_{\mathcal{U}}, ssk_{\mathcal{U}})$.
2. As in System One, in this step $\mathcal{U}$ and $\mathcal{B}$ contribute randomness to the wallet secret s, and the user selects wallet secrets $\boldsymbol{t} = (t_0, \ldots, t_{3\ell})$, where $t_i \in \mathbb{Z}_q$ for all i. As before, this is done as follows: $\mathcal{U}$ chooses random values s' and $\boldsymbol{t}$, and

sends the commitment $A' = \text{PedCom}(u, s', \boldsymbol{t})$ to $\mathcal{B}$, obtains a random r', sets $s = s' + r'$, and then both $\mathcal{U}$ and $\mathcal{B}$ locally set $A = g_2^{r'} A' = \text{PedCom}(u, s, \boldsymbol{t})$.

3. $\mathcal{U}$ forms a verifiable encryption Q of the value s under his own key $ek_{\mathcal{U}} = \mathsf{g}_3^u$. (This encryption can be proved correct relative to commitment A.) Q is signed by $\mathcal{U}$. $\mathcal{B}$ verifies the correctness of Q and the signature σ on Q. $\mathcal{U}$ obtains a CL signature from $\mathcal{B}$ on the values committed in A via the protocol for getting a signature on a set of committed values.
4. $\mathcal{B}$ debits 2^ℓ from account $pk_{\mathcal{U}}$, records the entry $(pk_{\mathcal{U}}, Q, \sigma)$ in his database D, and issues $\mathcal{U}$ a CL signature on Y.
5. $\mathcal{U}$ saves the wallet $W = (sk_{\mathcal{U}}, s, \boldsymbol{t}, \sigma_B(u, s, \boldsymbol{t}), J)$, where J is an ℓ-bit counter set to zero.

Spend$(\mathcal{U}(W, pk_{\mathcal{M}}), \mathcal{M}(sk_{\mathcal{M}}, pk_{\mathcal{B}}, 2^\ell))$**:** The only change from System One is in the calculation of the security tag T and the subsequent proof Φ. Assume $info \in \{0,1\}^*$ and $R \in \mathbb{Z}_q^*$ are obtained as before.

1. $\mathcal{U}$ sends the serial number of the coin $S = f_s^{DY}(J) = \mathsf{g}_2^{1/(J+s+1)}$, the security tag $T = pk_{\mathcal{U}} f_{\boldsymbol{t}}^V(J)^R = \mathsf{g}_3^{u+Rt_0 \prod_{\{i:V(J)_i=1\}} t_i}$, and a proof Φ of their validity to $\mathcal{M}$. The signature proof Φ consists of:

(a) $A_0 = \text{PedCom}(J)$ and a proof that A is a commitment to an integer in the range $[0, \ldots, 2^\ell - 1]$; a commitment $A_1 = \text{PedCom}(J^2)$ and a proof that it is a commitment to the square of the opening of A_0; and finally, a commitment $A_2 = \text{PedCom}(V(J)) = \text{PedCom}(J \circ J^2)$ and a proof that it was formed correctly.
(b) $B_i = \text{PedCom}(V(J)_i)$ for $i = 1$ to 3ℓ (commitments to the bits of $V(J)$) and proof that each B_i opens to either 0 or 1; that is, $PK\{(\gamma_1, \gamma_2) : B_i/g_1 = {h_1}^{\gamma_1} \vee B_i = {h_1}^{\gamma_2}\}$,
(c) proof that A_2 and $\{B_i\}$ are consistent; $PK\{(\gamma) : A_2/g_1 \prod_{i=1}^{3\ell} B_i^{2^{i-1}} = {h_1}^{\gamma}\}$,
(d) commitments to $\mathcal{U}$'s secret key u, and wallet secrets s and $\boldsymbol{t}$: $C = \text{PedCom}(u)$, $D = \text{PedCom}(s)$, $E_i = \text{PedCom}(t_i)$ for $i = 0$ to 3ℓ, and proof of knowledge of a CL signature on the openings of C, D, and all E_i's in that order,
(e) the following commitments that will help in proving that T was formed correctly: $F_0 = E_0$, $F_i = \text{PedCom}(\prod_{\{j \le i:V(J)_j=1\}} t_j)$ for $i = 1$ to 3ℓ,
(f) and a proof that $S = f_s^{DY}(J)$ and $T = \mathsf{g}_3^u (f_{\boldsymbol{t}}^V(J))^R$. Proving the statement about S is done as in System One; proving the statement about T can be done as follows: Prove, for every $1 \le i \le 3\ell$ that F_i was formed correctly, corresponding to the committed value t_i and the value contained in the commitment B_i. That is to say:

$$\begin{aligned} PK\{(\alpha, \beta, \delta) : {} & F_i = \text{PedCom}(\alpha) \ \wedge \ F_{i-1} = \text{PedCom}(\beta) \ \wedge \\ & E_i = \text{PedCom}(\delta) \ \wedge \ \Big(\big(B_i = \text{PedCom}(0) \ \wedge \ \alpha = \beta\big) \ \vee \\ & \big(B_i = \text{PedCom}(1) \ \wedge \ \alpha = \beta\delta\big) \Big) \} \end{aligned}$$

Note that, if all F_i's are formed correctly, then $F_{3\ell}$ is a commitment to the discrete logarithm of the value $f_{\boldsymbol{t}}^V(J) = t_0 \prod_{i=1}^{3\ell} t_i^{(V(J))_i}$. So we can prove the validity of tag T as follows:

$$PK\{(\alpha, \beta) \ : \ T = \mathsf{g}_3^{\alpha+\beta R} \wedge C = \mathrm{PedCom}(\alpha) \wedge F_{3\ell} = \mathrm{PedCom}(\beta)\}$$

2. $\mathcal{M}$ and $\mathcal{U}$ proceed exactly as before.

As in System One, using the Fiat-Shamir heuristic we turn these multiple proofs of knowledge into one signature, secure in the random-oracle model.

The Deposit protocol and VerifyGuilt algorithm follow the same outline as System One. During deposit, the bank may store only (S, R, T) in database L to obtain all desired functionality – except the ability to convince a third party of anything, such as a double-spender's identity or which coins belong to him. In the Identify protocol, the proof of guilt Π_G will additionally include the part of the user's secret key recovered as g_3^u.

Trace$(\zeta, S, pk_{\mathcal{U}}, \Pi_G, D, 2^\ell)$**:** Parse Π_G as (dk, π_1, π_2) and $pk_{\mathcal{U}}$ as $(ek_{\mathcal{U}}, vk_{\mathcal{U}})$. The bank checks that $e(\mathsf{g}_3, dk) = ek_{\mathcal{U}}$; if not, it aborts. Otherwise the bank searches its database D, generated during the withdrawal protocol, for verifiable encryptions tagged with the public key $pk_{\mathcal{U}}$. For each matching entry $(pk_{\mathcal{U}}, Q, \sigma)$, $\mathcal{B}$ does the following: (1) runs the Camenisch-Damgård decryption algorithm on Q with dk to recover the value s; and (2) then for $i = 0$ to $2^\ell - 1$, outputs a serial number $S_i = f_s^{DY}(i)$ and a proof of ownership $\Pi_i = (Q, \sigma, dk, i)$.

VerifyOwnership$(\zeta, S, \Pi, pk_{\mathcal{U}}, 2^\ell)$**:** Parse Π as (Q, σ, dk, i). Check that σ is $pk_{\mathcal{U}}$'s signature on Q and that i is in the range $[0, \ldots, 2^\ell - 1]$. Next, verify that dk is $pk_{\mathcal{U}}$'s decryption key by checking that $e(\mathsf{g}_3, dk) = ek_{\mathcal{U}}$. Finally, run the verifiable decryption algorithm on Q with dk to recover s' and verify that $S = f_{s'}^{DY}(i)$. If all checks pass, the algorithm accepts, otherwise, it rejects.

Efficiency Discussion of System Two. In Withdraw, the number of communication rounds does not change from System One, but one of the multi-base exponentiations will involve 3ℓ bases and hence its computation will take longer. Let us discuss the computational load of the verifiable encryption. For a cheating probability of at most 2^{-k}, the user must additionally compute k exps and $2k$ encryptions with the bilinear El Gamal scheme. To verify, the bank also must perform k exps but only k encryptions. Upon recovery of the double-spender's secret key, the bank needs to perform at most k decryptions and k exponentiations. Furthermore, the bank needs to compute all the 2^ℓ serial numbers each of which takes one exponentiation.

In Spend, the user must compute a total of $7 + 9\ell$ and $17 + 21\ell$ multi-base exponentiations for the commitments and the signature proof, respectively. The merchant and the bank also need to perform $17+21\ell$ multi-base exponentiations. For each of these, there is one multi-base exponentiation with 3ℓ exponents while all the others involve two to four bases.

Theorem 2. *System Two supports the algorithms* (BKeygen, UKeygen, Withdraw, Spend, Deposit, Identify, VerifyGuilt, Trace, VerifyOwnership) *and guarantees balance, identification of double-spenders, tracing of double-spenders, anonymity of users, and weak and strong exculpability under the Strong RSA, y-DDHI, and SF-DDH assumptions in the random oracle model.*

Proof of Theorem 2 appears in the full version of this paper [13]. Random oracles can be removed as discussed in System One.

Acknowledgments. We are grateful to Yevgeniy Dodis for helpful comments.

Part of Jan Camenisch's work reported in this paper is supported by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT and by the IST Project PRIME. The PRIME projects receives research funding from the European Community's Sixth Framework Programme and the Swiss Federal Office for Education and Science. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):591–610, 2000.
2. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In *NDSS '05*, pp. 29–43, 2005. Full version available at http://eprint.iacr.org/2005/028.
3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT '97*, vol. 1233, pp. 480–494, 1997.
4. D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, vol. 3027 of *LNCS*, pp. 54–73, 2004.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures using strong Diffie-Hellman. In *CRYPTO '04*, vol. 3152 of LNCS, pp. 41–55, 2004.
6. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT '00*, vol. 1807 of *LNCS*, pp. 431–444, 2000.
7. S. Brands. Electronic cash systems based on the representation problem in groups of prime order. In *Preproceedings of CRYPTO '93*, pp. 26.1–26.15, 1993.
8. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 318–333, 1997.
9. S. Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy.* PhD thesis, Eindhoven Inst. of Tech. The Netherlands, 1999.
10. E. Brickell, P. Gemmel, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *SIAM*, pp. 457–466, 1995.
11. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In T. Okamoto, editor, *ASIACRYPT '00*, vol. 1976 of *LNCS*, pp. 331–345, 2000.
12. J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *proceedings of SCN '04*, vol. 3352 of LNCS, pp. 120–133, 2004.

13. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash, 2005. Full version available at http://eprint.iacr.org/2005/060.
14. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, vol. 2576 of *LNCS*, pp. 268–289, 2003.
15. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, vol. 3152 of *LNCS*, pp. 56–72, 2004.
16. J. Camenisch and M. Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In *EUROCRYPT '99*, vol. 1592, pp. 107–122, 1999.
17. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO '99*, vol. 1666 of *LNCS*, pp. 413–430, 1999.
18. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO '03*, vol. 2729 of *LNCS*, pp. 126–144, 2003.
19. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, vol. 1296 of LNCS, pp. 410–424, 1997.
20. J. L. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998.
21. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 561–575, 1998.
22. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO '82*, pp. 199–203. Plenum Press, 1982.
23. D. Chaum. Blind signature systems. In *CRYPTO '83*, pp. 153–156. Plenum, 1983.
24. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.
25. D. Chaum. Online cash checks. In *EUROCRYPT '89*, vol. 434, pp. 289–293, 1989.
26. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO '90*, vol. 403 of *LNCS*, pp. 319–327, 1990.
27. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, vol. 740 of *LNCS*, pp. 89–105, 1993.
28. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, vol. 839 of *LNCS*, pp. 174–187, 1994.
29. I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, vol. 2501 of *LNCS*, 2002.
30. Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *Public Key Cryptography*, vol. 2567 of *LNCS*, pp. 1–17, 2003.
31. Y. Dodis and A. Yampolsky. A Verifiable Random Function with Short Proofs an Keys. In *Public Key Cryptography '05*, vol. 3386 of LNCS, pp. 416–431, 2005.
32. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86*, vol. 263 of LNCS, pp. 186–194, 1986.
33. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO '97*, vol. 1294 of *LNCS*, pp. 16–30, 1997.
34. S. Jarecki and V. Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In *EUROCRYPT*, vol. 3027 of *LNCS*, pp. 590–608, 2004.
35. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT '04*, vol. 3027 of LNCS, pp. 571–589, 2004.
36. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51, Number 2:231–262, 2004.
37. T. Okamoto and K. Ohta. Disposable zero-knowledge authentications and their applications to untraceable elec. cash. In *CRYPTO*, vol. 435, pp. 481–496, 1990.
38. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '92*, vol. 576 of *LNCS*, pp. 129–140, 1992.

39. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
40. M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In *EUROCRYPT '95*, vol. 921 of *LNCS*, pp. 209–219, 1995.
41. Y. S. Tsiounis. *Efficient Electonic Cash: New Notions and Techniques*. PhD thesis, Northeastern University, Boston, Massachusetts, 1997.

Cryptographic Asynchronous Multi-party Computation with Optimal Resilience (Extended Abstract)*

Martin Hirt[1], Jesper Buus Nielsen[2,**], and Bartosz Przydatek[1]

[1] Department of Computer Science, ETH Zurich,
8092 Zurich, Switzerland
{hirt, przydatek}@inf.ethz.ch
[2] Department of Computer Science, University of Aarhus,
DK-8200 Aarhus, Denmark
buus@daimi.au.dk

Abstract. We consider secure multi-party computation in the asynchronous model and present an efficient protocol with optimal resilience. For n parties, up to $t < n/3$ of them being corrupted, and security parameter κ, a circuit with c gates can be securely computed with communication complexity $\mathcal{O}(cn^3\kappa)$ bits. In contrast to all previous asynchronous protocols with optimal resilience, our protocol requires access to an expensive broadcast primitive only $\mathcal{O}(n)$ times — independently of the size c of the circuit. This results in a practical protocol with a very low communication overhead.

One major drawback of a purely asynchronous network is that the inputs of up to t honest parties cannot be considered for the evaluation of the circuit. Waiting for all inputs could take infinitely long when the missing inputs belong to corrupted parties. Our protocol can easily be extended to a hybrid model, in which we have one round of synchronicity at the end of the input stage, but are fully asynchronous afterwards. In this model, our protocol allows to evaluate the circuit on the inputs of every honest party.

1 Introduction

Secure multi-party computation. The goal of secure multi-party computation (MPC) is to allow a set of n players to evaluate an agreed function of their inputs in a secure way, where security means that an adversary corrupting some of the players cannot achieve more than controlling the inputs and outputs of these players. In particular, the adversary does not learn the inputs of the uncorrupted players, and furthermore, she cannot influence the outputs of the uncorrupted players except by selecting the inputs of the corrupted players.

* The full version of this paper is available at *Cryptology ePrint Archive* [HNP04].
** Supported by the Danish National Science Research Council grant No. 21-02-0093.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 322–340, 2005.

We consider a static active t-adversary who can corrupt up to t of the players and take full control over them. Furthermore, we focus on *asynchronous communication*, i.e., the messages in the network can be delayed for an arbitrary amount of time (but eventually, all messages are delivered). As a worst-case assumption, we give the ability of controlling the delay of messages to the adversary.

Asynchronous communication models real-world networks (like the Internet) much better than synchronous communication. However, it turns out that MPC protocols for asynchronous networks are significantly more involved than their synchronous counterparts. One reason for this is that in an asynchronous network, when a player does not receive an expected message, he cannot distinguish whether the sender is corrupted and did not send the message, or the message was sent but delayed in the network. This implies also that in a fully asynchronous setting it is impossible to consider the inputs of *all* uncorrupted players when evaluating the function. The inputs of up to t (potentially honest) players have to be ignored, because waiting for them could turn out to be endless [Bec54].

History and related work. The MPC problem was first proposed by Yao [Yao82] and solved by Goldreich, Micali, and Wigderson [GMW87] for computationally bounded adversaries and by Ben-Or, Goldwasser, and Wigderson [BGW88] and independently by Chaum, Crépeau, and Damgård [CCD88] for computationally unbounded adversaries. All these protocols considered a synchronous network with a global clock. The first MPC protocol for the asynchronous model (with unconditional security) was proposed by Ben-Or, Canetti, and Goldreich [BCG93]. Extensions and improvements, still in the unconditional model, were proposed in [BKR94, SR00, PSR02]. A great overview of asynchronous MPC with unconditional security is given in [Can95].

The most efficient asynchronous protocols up to date are the ones of Srinathan and Rangan [SR00] and of Prabhu, Srinathan and Rangan [PSR02]. The former protocol requires $\Omega(n^2)$ invocations to the broadcast primitive for every multiplication, which makes the protocol very inefficient when broadcast is realized with some asynchronous broadcast protocol. The latter protocol is rather efficient; it requires $\Omega(n^4\kappa)$ bits of communication per multiplication. However, it tolerates only $t < n/4$ corruptions, which is non-optimal.

Contributions. We present the first asynchronous MPC protocol for the cryptographic model. The protocol is secure with respect to an active adversary corrupting up to $t < n/3$ players; this is optimal in an asynchronous network.

The main achievement of the new protocol is its efficiency: Once the inputs are distributed, the protocol requires $\mathcal{O}(c_M n^3 \kappa)$ bits of communication to evaluate a circuit with c_M multiplication gates and with security parameter κ. This is the same communication complexity that is required by the most efficient known protocol for the synchronous model [CDN01], and improves on the communication complexity of the most efficient optimally-secure asynchronous MPC protocol [SR00] by a factor of $\Omega(n)$. In contrast to both the protocols of [CDN01] and [SR00], our protocol uses broadcast only in a very limited manner: the number of broadcast invocations is independent of the size of the circuit. This

nice property is also achieved in [PSR02], but this protocol is non-optimal (it tolerates only $t < n/4$) and requires $\Omega(n)$ times more communication than ours.

In an asynchronous MPC, the agreed function can be evaluated only on a subset of the inputs, i.e., some (potentially honest) player cannot provide their input into the computation. However, the presented protocol can easily be extended to consider the input of each (honest) party, at the cost of one round of synchronization required at the end of the input stage.

2 Preliminaries, Notation and Tools

2.1 Formal Model

We use the model of security of asynchronous protocols from [Can01]. Formally our model for running a protocol will be the hybrid model with a functionality for distributing some initial cryptographic keys between the parties using some function init. The ideal functionality that we wish to realize is given by a circuit Circ, or more precisely a family of circuits. Namely, the functionality allows the adversary to specify a set of at least $n - t$ parties, $W \subseteq [n]$ (where $[n]$ denotes the set $\{1, \ldots, n\}$), which are to supply the inputs to the computation. The circuit to be computed, $\mathsf{Circ} = \mathsf{Circ}(W)$, is then uniquely defined by the subset of parties providing the inputs. The informal proofs in this extended abstract do not require familiarity with specific details of the model in [Can01], and below we only recall the needed specificities.

Asynchronous protocols. An n-player protocol is a tuple $\pi = (P_1, \ldots, P_n, \mathsf{init})$, where each P_i is a probabilistic interactive Turing machine, and init is an *initialization function*, used for the usual set-up tasks (initialize the players, set up cryptographic keys, etc.). The parties (players) communicate over an *asynchronous* network, in which the delay between sending and delivery of a message is unbounded. More precisely, when a party sends a message, this message is added to the set of messages already sent but not yet delivered, $\mathsf{Msg} = \{(i, j, m)\}$, where (i, j, m) denotes a message m from P_i to P_j. The delivery of the messages is scheduled by the adversary (see below).

We assume that the function to be computed is given as a circuit consisting of input gates augmented by the party to supply the input, linear gates and multiplication gates, and output gates augmented by the party to see the output, all over some ring $\mathcal{M}$.

Adversary. We consider a polynomially bounded adversary, and our constructions are parametrized by a security parameter κ. The adversary controls the delivery of messages and can corrupt up to t parties. A corrupted party is under full control of the adversary, which sees all incoming messages, and determines all outgoing messages. The adversary schedules the delivery of the messages arbitrarily, by picking a message $(i, j, m) \in \mathsf{Msg}$ and delivering it to the recipient. The adversary doesn't see the contents of messages exchanged between honest (i.e., not corrupted) parties, and any message from an honest party to an honest party is *eventually* delivered. In most cases we require that $t < n/3$, but will sometimes

consider other thresholds. The set of parties to be corrupted is specified by the adversary *before* the execution of the protocol, i.e., we consider static security.

Execution of a protocol. Before the protocol starts, an initialization function init is evaluated on the security parameter 1^κ and on random input $r \in \{0,1\}^*$, to generate a tuple $(\mathrm{sv}_1, \ldots, \mathrm{sv}_n, \mathrm{pv}) = \mathsf{init}(\kappa, r)$ of secret values sv_i and a public value pv. Each party P_i is initialized with $(1^\kappa, \mathrm{sv}_i, \mathrm{pv})$. At the beginning of the protocol execution, every party P_i receives its input value x_i from the environment, and produces some initial messages $(i, \cdot, \cdot)$ which are added to the set Msg. The adversary is given the public value pv, the values (x_j, sv_j) for each corrupted party P_j, and the control over the set Msg. Subsequently the protocol is executed in a sequence of *activations*. In each activation the adversary picks a message $(i, j, m) \in \mathsf{Msg}$ and delivers it to P_j. Upon delivery of a message, party P_j performs some computation based on its current state, updates its state and produces some messages of the form $(j, \cdot, \cdot)$, which are added to the set Msg. In some activation the parties can produce the output to the environment and terminate. The adversary determines the inputs x_i and all messages of corrupted parties. The adversary and the environment can communicate with each other.

Security. The security of a protocol is defined relative to an ideal evaluation of the circuit by requiring that for any adversary attacking the execution of the protocol there exists a simulator which can simulate the attack of the adversary to any environment given only an ideal process for evaluating the circuit. In the ideal process the simulator has very restricted capabilities: It sees the inputs of the corrupted parties. Then it specifies a subset $W \subseteq [n]$ of the parties to be the input providers, under the restriction that $|W| \geq n - t$. The set W is used to pick the circuit $\mathsf{Circ} = \mathsf{Circ}(W)$ to be evaluated. The input gates of Circ are assigned the inputs of the corresponding parties (the adversary specifies the inputs of the corrupted parties), then Circ is evaluated and the outputs of the corrupted parties are shown to the simulator. Given these capabilities the simulator must then simulate to the environment the entire view of an execution of the protocol, including the messages sent and the possible communication between the environment and the adversary.

In the following subsections we briefly describe cryptographic tools needed in our constructions, and introduce a notation for their use in the rest of the paper.

2.2 Homomorphic Public-Key Encryption with Threshold Decryption

We assume the existence of a semantically secure probabilistic public-key encryption scheme, which also is homomorphic and enables threshold decryption:

Encryption and decryption. For an encryption key e and a decryption key d, let $\mathscr{E}_e : \mathcal{M} \times \mathcal{R} \to \mathcal{C}$ denote the encryption function mapping (plaintext, randomness) pair $(c, r) \in \mathcal{M} \times \mathcal{R}$ to a ciphertext $C \in \mathcal{C}$, and let $\mathscr{D}_d : \mathcal{C} \to \mathcal{M}$ denote the corresponding decryption function. We require that $\mathcal{M}$ is a ring $\mathbb{Z}_M$ for some $M > 1$, and we use $\cdot$ to denote multiplication in $\mathcal{M}$. We often use

capital letters to denote an encryption of the corresponding lower-case letter. When keys are understood, we write $\mathscr{E}$ and $\mathscr{D}$ instead of $\mathscr{E}_e$ resp. $\mathscr{D}_d$, and we frequently omit the explicit mention of the randomness in encryption function $\mathscr{E}$.

Homomorphic property. We require that there exist (efficiently computable) binary operations $+$, $*$, and $\oplus$, such that $(\mathcal{M}, +)$, $(\mathcal{R}, *)$, and $(\mathcal{C}, \oplus)$ are algebraic groups, and that $\mathscr{E}_e$ is a group homomorphism, i.e. that

$$\mathscr{E}(a, r_a) \oplus \mathscr{E}(b, r_b) = \mathscr{E}(a + b, r_a * r_b) \ .$$

We use $A \ominus B$ to denote $A \oplus (-B)$, where $-B$ denotes the inverse of B in the group $\mathcal{C}$. For an integer a and $B \in \mathcal{C}$ we use $a \cdot B$ to denote the sum of B with itself a times in $\mathcal{C}$.

Ciphertext re-randomization. For $C \in \mathcal{C}$ and $r \in \mathcal{R}$ we let $\mathscr{R}_e(C, r) = C \oplus \mathscr{E}_e(0, r)$. We use $C' = \mathscr{R}_e(C)$ to denote $C' = \mathscr{R}_e(C, r)$ for uniformly random $r \in \mathcal{R}$. We call $C' = \mathscr{R}_e(C)$ a re-randomization of C. Note that C' is a uniformly random encryption of $\mathscr{D}_d(C)$.

Threshold decryption. We require that there exists a threshold function sharing of $\mathscr{D}_d$ among n parties, i.e., for some construction threshold $1 < t_{\mathsf{D}} \leq n$ there exists a sharing $(d_1, \ldots, d_n)$ of the decryption key d (where d_i is intended for party P_i), such that given decryption shares $c_i = \mathscr{D}_{i,d_i}(C)$ for t_{D} distinct decryption-key shares d_i, it is possible to efficiently compute c such that $c = \mathscr{D}_d(C)$. Furthermore, the encryption scheme should be still semantically secure against chosen plaintext attack when the adversary is given $t_{\mathsf{D}} - 1$ decryption-key shares. Finally, we require that given a ciphertext C, plaintext $c = \mathscr{D}_d(C)$, and a set of $t_{\mathsf{D}} - 1$ decryption-key shares $\{d_i\}$, it is possible to compute *all* n decryption shares $c_j = \mathscr{D}_{j,d_j}(C)$, $j = 1, \ldots, n$. We will always have $t_{\mathsf{D}} = t + 1$. When keys are understood, we write $\mathscr{D}_i(C)$ to denote the function computing decryption share of party P_i for ciphertext C, and $c = \mathscr{D}(C, \{c_i\})$ to denote the process of combining the decryption shares $\{c_i\}$ to a plaintext c.

Robustness. To efficiently protect against cheating servers we require that there exists an efficient two-party zero-knowledge protocol for proving the correctness of a decryption share $c_i = \mathscr{D}_{i,d_i}(C)$ given (e, C, c_i) as instance, and given (i, d_i) and randomness r as witness. We require also that there exists an efficient two-party zero-knowledge protocol for proving the knowledge of a plaintext, given (e, C) as instance and the corresponding plaintext c and randomness r as witness. We require that these protocols communicate $\mathcal{O}(\kappa)$ bits per proof.

2.3 Digital Signatures

We assume the existence of a digital signature scheme unforgeable against an adaptive chosen message attack. For a signing key s and a verifying key v, let $\mathsf{Sign}_s : \{0,1\}^* \to \{0,1\}^\kappa$ denote the signing function, and let $\mathsf{Ver}_v : \{0,1\}^* \times \{0,1\}^\kappa \to \{0,1\}$ denote the verifying function, where $\mathsf{Ver}_v(m, \sigma) = 1$ indicates that σ is a valid signature on m. We write $\mathsf{Sign}_i/\mathsf{Ver}_i$ to denote the signing/verification operation of party P_i.

2.4 Threshold Signatures

We assume the existence of a *threshold* signature scheme unforgeable against an adaptive chosen message attack. For a signing key s and a verifying key v, let $\mathscr{S}_s : \{0,1\}^* \to \{0,1\}^\kappa$ denote the signing function, and let $\mathscr{V}_v : \{0,1\}^* \times \{0,1\}^\kappa \to \{0,1\}$ denote the verifying function, where $\mathscr{V}_v(m,\sigma) = 1$ indicates that σ is a valid signature on m.

Threshold signing. We require that there exists a threshold function sharing of $\mathscr{S}_s$ among n parties, i.e., for some signing threshold $1 < t_{\mathsf{S}} \leq n$ there exists a sharing $(s_1, \ldots, s_n)$ of the signing key s (where s_i is intended for party P_i), such that given signature shares $\sigma_i = \mathscr{S}_{i,s_i}(m)$ for t_{S} distinct signing-key shares s_i, it is possible to efficiently compute σ such that $\mathscr{V}_v(m,\sigma) = 1$. Furthermore, the threshold signature scheme should be still unforgeable against adaptive chosen message attack when the adversary is given $t_{\mathsf{S}} - 1$ signing-key shares. Finally, we require that given a signature σ on m, and $t_{\mathsf{S}} - 1$ signing-key shares $\{s_i\}$, it is possible to compute *all* n signature shares $\sigma_j = \mathscr{S}_{j,s_j}(m)$, $j = 1, \ldots, n$. We will always have $t_{\mathsf{S}} = n - t$. When keys are understood, we write $\mathscr{S}_i(m)$ to denote the function computing signature share of party P_i for message m, and $\sigma = \mathscr{S}(m, \{\sigma_i\})$ to denote the process of combining the signature shares $\{\sigma_i\}$ to a signature σ.

Robust threshold signing. To efficiently protect against cheating servers we require that there exists an efficient two-party zero-knowledge protocol for proving the correctness of a signature share $\sigma_i = \mathscr{S}_{i,s_i}(m)$, given (v, m, σ_i) as instance and given (i, s_i) as witness. We require that this protocol communicates $\mathcal{O}(\kappa)$ bits per proof.

2.5 Byzantine Agreement

We assume the existence of a Byzantine agreement (BA) protocol, i.e., a protocol with the following properties: The input of party P_i is a bit $v_i \in \{0,1\}$ and the output of the BA is a bit $w \in \{0,1\}$. If all honest parties enter the BA, then the BA eventually terminates. Furthermore, if the BA terminates with output w, then some honest party entered the BA with input $v_i = w$. In particular, if all honest parties have the same input $v_i = v$, then the output of the BA is $w = v$.

2.6 Cryptographic Assumptions and Instantiations of Tools

The security of our constructions is based on *decisional composite residuosity assumption* (DCRA) [Pai99]. Alternatively, it could be based also on QRA and strong RSA. We stress, that our constructions are in the plain model. In particular, our constructions *do not* make use of random oracles.

Homomorphic encryption with threshold decryption. An example of a scheme satisfying all required properties is Paillier's cryptosystem [Pai99] enhanced by threshold decryption as in [FPS00, DJ01]. In this scheme $\mathcal{M} = \mathbb{Z}_N$ for an RSA modulus N. Another example can be based on the QR assumption and the strong RSA assumption. [CDN01].

Digital signatures. As our digital signature scheme we use standard RSA signatures [RSA78].

Threshold signatures. As an example we can use the threshold signature scheme by Shoup [Sho00]. The security of the threshold signature scheme in [Sho00] is based on the assumption that standard RSA signatures are secure. As presented in [Sho00] the zero-knowledge proofs are non-interactive but for the random-oracle model. The protocol can be modified to be secure in the plain (random oracle devoid) model by using interactive proofs (cf. [Nie02]).

Byzantine agreement. In our protocols we employ the efficient Byzantine agreement protocol of Cachin *et al.* [CKS00], which has expected constant round complexity, and expected bit complexity of $\mathcal{O}(n^2\kappa)$. As presented in [CKS00] the security proof of the protocol needs the random-oracle methodology (for the above mentioned threshold signature scheme). This protocol also can be modified to be secure in the plain model [Nie02].

3 The New Protocol

Our protocol follows the paradigm of secure multi-party computation based on a threshold homomorphic encryption scheme, as introduced by Franklin and Haber [FH96], and made robust by Cramer, Damgård and Nielsen [CDN01]. However, both protocols use synchrony in an essential manner.

A high-level overview. The protocol proceeds in three stages, an *input stage*, an *evaluation stage*, and a *termination stage*. In the following, we briefly summarize the goal of each stage:

- *Input stage*: Every player provides an encryption of his input to every other player, and the players jointly agree on a subset of players who have correctly provided their inputs.
- *Evaluation stage*: Every player independently evaluates the circuit on a gate-by-gate basis, with help of the other players. The circuit consists of linear gates, multiplication gates, and output gates. The circuit may depend on the selected subset of players that have provided input.
- *Termination stage*: As soon as a player has completed the circuit evaluation, he moves into the termination stage, where the players jointly agree that the circuit evaluation is completed, every player has received (or will eventually receive) the output(s), and hence every player who is still in the evaluation stage can safely abort it.

By having *every* player evaluate the circuit on his own, we bypass the inherent synchronicity problems of the asynchronous model. We denote the player that evaluates the circuit as the *king*, and all other players (who support the king) the *slaves*. Note that every player acts (in parallel) once as king, and n times as slave, once for every king. The kings are not synchronized among each other; it can happen that one king has almost completed the evaluation of the circuit, while another king is still at the very beginning. However, each slave is synchronized

with his king. As soon as the first king completes the evaluation and provably reveals all outputs, all other kings (and their slaves) can safely truncate their own evaluation.

In order to achieve robustness, we must require every party to prove (in zero-knowledge) the correctness of essentially every value she provides during the protocol execution. These proofs could easily be constructed in the random-oracle model (by using Fiat-Shamir heuristics), but this would be at the costs of a non-standard model. We therefore follow another approach: A party who is to prove some claim, proves this claim interactively to every other party. The verifying party then certifies that she has correctly verified the claim. Once the prover has collected enough such certificates, she can convince any third party non-interactively of the validity of the claim. Technically, we use threshold signatures for the certificates, which allows the prover to compute one short certificate that proves that t_{S} parties have verified his proof (recall that t_{S} denotes the threshold of the threshold signature scheme, and we set $t_{\mathsf{S}} = n - t$). Formally, we will say that "a party P_i constructs and sends a proof π_i of «*some claim*»", denoted as $\pi_i = \mathsf{proof}(\text{«}\textit{some claim}\text{»})$, when we mean that P_i bilaterally proves the claim in zero-knowledge to every party P_j, who then, upon successful completion of the proof, sends to P_i a signature share $\mathscr{S}_j(\text{«}\textit{some claim}\text{»})$. After obtaining t_{S} correct signature shares[1] $\{\pi_{i,j}\}_{j\in I}$, party P_i computes π_i as $\mathscr{S}(\text{«}\textit{some claim}\text{»}, \{\pi_{i,j}\}_{j\in I})$. Since this construction is standard, we omit the details from the description of the protocols. Naturally, we do include the corresponding subprotocols and their bit complexities in the analysis of the proposed solution (cf. Section 3.8).

Finally we note that we make use of threshold signatures also explicitly, as specified in the descriptions of the protocols. Their use there has similar purpose, namely as certificates for the validity of certain claims.

3.1 Main Protocol

The main protocol first invokes the input stage, then the evaluation stage, and finally the termination stage. At the end of the input stage, the circuit Circ is determined by the set of parties that provide input. In the evaluation stage, every party starts one instance of the king protocol, and n instances of the slave protocol — one for every king.

In order to precisely describe the new protocol, we first formally model the circuit to be computed, and then give the invariant that is satisfied during the whole computation. For the clarity of presentation we assume in the following that every party provides at most one input value and that there is only one final output value to be disclosed to all parties (i.e., the final output is to be public). This is without loss of generality for the case with public outputs, and protocols for the general case with multiple input/output values can be derived by straightforward modifications. However, the issue of providing *private* outputs is more involved, and we discuss it in Section 4.1.

[1] The correctness of a signature share is proved to P_i by P_j, using another efficient zero-knowledge protocol (cf. Sect. 2.4).

THE CIRCUIT. We assume that the function to be computed is given as a circuit Circ over the plaintext space $\mathcal{M}$ of the homomorphic encryption scheme in use. The circuit is a set of labeled gates, where each label is a unique bit-string $G \in \{0,1\}^*$. We use $G(\mathsf{Circ})$ to denote the set of all labels of the circuit Circ. In the following we let $v : G(\mathsf{Circ}) \to \mathcal{M} \cup \{\perp\}$ be a map from the labels into the the plaintext space, where $v(G)$ denotes the value of the gate G. Each gate is represented by a tuple $(G, \ldots)$, and can have one of the following types:

input gate: (G), consisting only of its label $G = (P_i, \mathsf{input})$, where $v(G)$ is equal to x_i, the input value provided by player P_i.

linear gate: $(G, \mathsf{linear}, a_0, a_1, G_1, \ldots, a_l, G_l)$, where $l \geq 0$, $a_0, \ldots, a_l \in \mathcal{M}$ are constants, and $v(G) = a_0 + \sum_{j=1}^{l} a_j \cdot v(G_j)$.

multiplication gate: $(G, \mathsf{mul}, G_1, G_2)$, where $v(G) = v(G_1) \cdot v(G_2)$.

output gate: (G, output, G'), where $v(G) = v(G')$ is the output value of the circuit.

CORRECTNESS INVARIANT. Throughout the computation each party P_i maintains a data structure containing the views of each party P_j on the intermediate values in the circuit. More precisely, P_i holds a dictionary Γ_i, which for each party P_j maps labels G to encryptions computed by the King P_j,

$$\Gamma_i : [n] \times G(\mathsf{Circ}) \to \mathcal{C} \cup \{\perp\} \ ,$$

where initially $\Gamma_i(j, G) = \perp$ for all labels and all $j \in [n]$. If $\Gamma_i(j, G) = C \neq \perp$, then from P_i's point of view gate G was completed by P_j, and C is a ciphertext of the homomorphic encryption scheme of the value $v(G)$. We say that C is the encryption of $v(G)$ *reported* by P_j to P_i, and that P_i has *accepted* C from P_j.

The protocol guarantees, that if an honest party P_i accepts a ciphertext C reported by P_j, then C is an encryption of a correct value for gate G. Moreover, any two honest parties who accept an encryption of any party P_l for a gate G agree on the encryption. Formally, we have the following definition.

Definition 1 (Correctness Invariant). *The following properties hold at any point in the protocol:*

1. (Agreement on circuit) *There exists a set $W \subseteq [n]$ such that $|W| \geq n - t$ and such that for all honest parties having defined the circuit Circ_i it holds that $\mathsf{Circ}_i = \mathsf{Circ}(W)$.*
2. (Agreement on input encryptions) *For all pairs of honest parties P_i, P_j, and all $k, l, m \in W$ it holds that if $\Gamma_i(k, (P_l, \mathsf{input})) \neq \perp$ and $\Gamma_j(m, (P_l, \mathsf{input})) \neq \perp$, then*
$$\Gamma_i(k, (P_l, \mathsf{input})) = \Gamma_j(m, (P_l, \mathsf{input})) := X_l \ .$$
Furthermore, if P_l is honest then $\mathscr{D}(X_l)$ is the initial input x_l of P_l.
3. (Correct gate encryption) *For every honest party P_i, if for any $G \in \mathsf{Circ}$ we have that $\Gamma_i(i, G) = C \neq \perp$, then $\mathscr{D}(C)$ is identical to the value of gate G obtained by decrypting the input encryptions held by P_i and evaluating the circuit on the plaintexts.*

4. (Agreement on encryptions of gates by same king) *For every two honest parties P_i and P_j, for any $l \in [n]$ and any $G \in \mathsf{Circ}$, if $\Gamma_i(l, G) = C \neq \perp$ and $\Gamma_j(l, G) = C' \neq \perp$, then $C = C'$.*

This invariant is propagated from the initial input stage until the output stage is reached. Hence, a threshold decryption of the encrypted output value is guaranteed to yield correct computation results.

The basis of the correctness invariant. The correctness invariant is established in the input stage, which determines the values of all input gates. Due to the security properties of the input-stage protocol, these values are guaranteed to be *correct* in the sense that the each party providing input knows the actual value hidden in the encryption, and that this value is a valid input to the function to be computed.

3.2 Input Stage

The goal of the input stage is to define an encryption of the input of each party. To ensure independence of the inputs, the parties are required to prove plaintext knowledge for their encryptions. In a synchronous network we could simply let the parties broadcast their encryptions. However, in an asynchronous setting with an active adversary we cannot guarantee that each party contributes an input value, since it is impossible to distinguish between an honest slow party and a corrupted party. Therefore a protocol is used which selects $(n-t)$ so-called *input providers*.

First, each party P_i encrypts its input value x_i to obtain a ciphertext $X_i = \mathscr{E}(x_i)$, and constructs a proof $\pi_i = \mathsf{proof}(\text{«}P_i \textit{ knows the plaintext in } X_i\text{»})$ (using bilateral zero-knowledge proofs and threshold signatures), which serves as a certificate that P_i knows the encrypted value, and that X_i is P_i's unique possible input encryption to the circuit. Afterwards P_i distributes (X_i, π) to all parties, and then constructs and distributes another certificate, a *certificate of distribution* cert_i, stating that P_i has distributed (X_i, π_i) to at least $n-t$ parties (recall that $n-t$ is the threshold of the signature scheme).

When a party collects $n-t$ certificates of distributions it knows that at least $n-t$ parties have their certified inputs distributed to at least $n-t$ parties. So, at least $n-t$ parties had its certified input distributed to at least $(n-t)-t \geq t+1$ *honest* parties. So, if all honest parties echo the certified inputs they saw and collect $n-t$ echoes, then all honest parties will end up holding the certified input of the $n-t$ parties which had their certified inputs distributed to at least $t+1$ honest parties. These $n-t$ parties will eventually be the input providers. To determine who they are, n Byzantine agreements are run. The protocol for selecting input providers is given in more detail in Figure 1.

3.3 Computing Linear Gates

Due to the homomorphic property of encryption linear gates can be computed locally, without interaction. That is, if a party P_i has accepted P_j's encryptions

To define an initial set of inputs, P_i with initial input $x_i \in \mathcal{M}$ proceeds as follows:
Initialize empty sets $A_i, \boldsymbol{\mathcal{A}}_i, B_i, \boldsymbol{\mathcal{B}}_i, C_i$ and execute the following rules concurrently:

Double Distribution:

1. compute $X_i := \mathscr{E}(x_i)$ and $\pi_i := \mathsf{proof}(\text{«}P_i \textit{ knows the plaintext in } X_i\text{»})$.
2. send (X_i, π_i) to all parties.
3. collect $n-t$ signature shares $\{\sigma_j\}$ on «X_i *is* P_i*'s input*»
   (i.e., $\sigma_j = \mathscr{S}_j(\text{«}X_i \textit{ is } P_i\textit{'s input}\text{»})$)
4. compute $\mathsf{cert}_i = \mathscr{S}(\text{«}X_i \textit{ is } P_i\textit{'s input}\text{»}, \{\sigma_j\})$; send (X_i, cert_i) to all parties.
5. collect $n-t$ signature shares $\{\sigma'_j\}$ on «*I hold* P_i*'s input*»
6. compute $\mathsf{cert}'_i = \mathscr{S}(\text{«}I \textit{ hold } P_i\textit{'s input}\text{»}, \{\sigma'_j\})$; send cert'_i to all parties.

Grant Certificate of Uniqueness:

on the first msg. (X_j, π_j) from P_j, with $\mathscr{V}(\text{«}P_j \textit{ knows the plaintext in } X_j\text{»}, \pi_j) = 1$,
return $\sigma_i = \mathscr{S}_i(\text{«}X_j \textit{ is } P_j\textit{'s input}\text{»})$ to P_j.

Grant Certificate of Distribution:

on the first message (X_j, cert_j) from P_j, with $\mathscr{V}(\text{«}X_j \textit{ is } P_j\textit{'s input}\text{»}, \mathsf{cert}_j) = 1$,
add j to A_i, add (X_j, cert_j) to $\boldsymbol{\mathcal{A}}_i$, and return $\sigma'_i := \mathscr{S}_i(\text{«}I \textit{ hold } P_j\textit{'s input}\text{»})$ to P_j.

Echo Certificate of Distribution:

on a message cert'_j, where $\mathscr{V}(\text{«}I \textit{ hold } P_j\textit{'s input}\text{»}, \mathsf{cert}'_j) = 1$ and $j \notin C_i$,
add j to C_i, and send cert'_j to all parties.

Select Input Providers:

When $|C_i| \geq n-t$, stop executing the above rules and proceed as follows:

1. send $(A_i, \boldsymbol{\mathcal{A}}_i)$ to all parties.
2. collect a set $\{(A_j, \boldsymbol{\mathcal{A}}_j)\}_{j \in J}$ of $(n-t)$ incoming, well-formed $(A_j, \boldsymbol{\mathcal{A}}_j)$.
3. let $B_i := \bigcup_{j \in J} A_j$ and $\boldsymbol{\mathcal{B}}_i := \bigcup_{j \in J} \boldsymbol{\mathcal{A}}_j$
4. enter n Byzantine Agreements (BAs) with inputs $v_1, \ldots, v_n \in \{0,1\}$,
   where $v_j = 1$ iff $j \in B_i$.
5. let $w_1, \ldots, w_n$ denote the outputs of the BAs,
   and let $W = \{j \in \{1, \ldots, n\} | w_j = 1\}$.
6. use W to generate a circuit $\mathsf{Circ} = \mathsf{Circ}(W)$.
7. for each $j \in B_i \cap W$, send $(X_j, \mathsf{cert}_j) \in \boldsymbol{\mathcal{B}}_i$ to all parties.
8. for each $j \in W$ wait to receive (X_j, cert_j) with $\mathscr{V}(\text{«}X_j \textit{ is } P_j\textit{'s input}\text{»}, \mathsf{cert}_j) = 1$.
9. for all $j \in W$ and $l \in \{1, \ldots, n\}$, let $\Gamma_i(l, (P_j, \mathsf{input})) = X_j$.

Fig. 1. The input stage code for P_i

of inputs to a linear gate $(G, \mathsf{linear}, a_0, G_1, a_1, \ldots, G_l, a_l)$, i.e. when $\Gamma_i(j, G_u) \neq \perp$, for $u = 1 \ldots l$, then P_i computes locally $\Gamma_i(j, G) := A_0 \oplus \left(\bigoplus_{u=1}^{l} (a_j \cdot \Gamma_i(j, G_u)) \right)$, where A_0 is a *"dummy"* encryption of a_0, computed using a publicly-known, fixed random string.

Wait until input stage is completed, resulting in a circuit Circ *and an initialized dictionary* Γ_k*. Then concurrently execute for each linear, multiplication, or output gate:*

LINEAR GATE $(G, \mathsf{linear}, a_0, a_1, G_1, \ldots, a_l, G_l)$:

1. wait until $\Gamma_k(k, G_u) \neq \perp$, for all $u = 1 \ldots l$.
2. compute $\Gamma_k(k, G) := A_0 \oplus \left(\bigoplus_{u=1}^{l} (a_u \cdot \Gamma_k(k, G_u)) \right)$.

MULTIPLICATION GATE $(G, \mathsf{mul}, G_1, G_2)$:

1. wait until $\Gamma_k(k, G_1) = C_1 \neq \perp$ and $\Gamma_k(k, G_2) = C_2 \neq \perp$
2. generate a randomizer (R, U, cert) for G, and send it to all parties:
   (a) collect a set $S_G := \{(R_i, U_i, \sigma_i, \pi_i)\}_{i \in I_G}$, with $|I_G| \geq t + 1$, where
   $\sigma_i = \mathsf{Sign}_i(\langle\!\langle (R_i, U_i) : \textit{part of } P_k\textit{'s randomizer for } G \rangle\!\rangle)$, and
   $\pi_i = \mathsf{proof}_i(\langle\!\langle P_i \textit{ knows } r_i \textit{ in } R_i\textit{, and } U_i \textit{ is a randomization of } r_i C_1 \rangle\!\rangle)$
   (b) send S_G to all parties
   (c) compute $R := \bigoplus_{i \in I_G} R_i$, and $U := \bigoplus_{i \in I_G} U_i$
   (d) collect a set $\mathsf{cert} := \{\mathsf{cert}_i\}_{i \in I'_G}$, with $|I'_G| \geq t_{\mathsf{S}}$, where
   $\mathsf{cert}_i = \mathscr{S}_i(\langle\!\langle (R, U) : P_k\textit{'s randomizer for } G \rangle\!\rangle)$
3. collect a set $V_G = \{(z_i, \phi_i)\}_{i \in I''_G}$, with $|I''_G| \geq t_{\mathsf{D}}$, where each z_i is a decryption share of P_i for $Z = C_2 \oplus R$, and ϕ_i is the corresponding validity proof
4. send V_G to all parties
5. decrypt $z := \mathscr{D}(Z, V_G)$ and compute $\Gamma_k(k, G) := (z \cdot C_1) \ominus U$

OUTPUT GATE (G, output, G'):

1. wait until $\Gamma_k(k, G') = C \neq \perp$
2. collect a set $\{(c_i, \varpi_i)\}$ of t_{D} decryption shares for C, with corresponding validity proofs ϖ_i
3. compute and output $c := \mathscr{D}(C, \{c_i\})$; mark G as decrypted

Fig. 2. The code for king P_k for evaluating the circuit

3.4 Computing Multiplication Gates

Computation of multiplication gates is more involved. Each king P_k leads the computation of the encrypted product in *his* copy of the circuit, that is, given a gate $(G, \mathsf{mul}, G_1, G_2)$ such that $\Gamma_k(k, G) = \perp$, $\Gamma_k(k, G_1) = C_1$, and $\Gamma_k(k, G_2) = C_2$, with $C_1, C_2 \neq \perp$, the players proceed as follows. Let c_1, c_2 denote the values hidden in the ciphertexts C_1, C_2, respectively. First a randomizer (R, U, cert) is generated, where R is a threshold encryption of a random element $r \in \mathcal{M}$ (unknown to the parties and the adversary), $U = \mathscr{R}(rC_1)$, i.e., U is a random threshold encryption of rc_1, and cert is a certificate of the encryptions' correctness. Then P_k sends the randomizer to all parties, and waits until the parties answer with decryption shares of the ciphertext $Z = C_2 \oplus R$, which is an encryption of $z = c_2 + r$. Once sufficiently many (i.e., at least t_{D}) decryption shares arrive, P_k sends them to all parties, which allows each P_i to decrypt z, and compute an encryption of the product $c_1 c_2$, using the homomorphic property of

Wait until input stage is completed, resulting in a circuit Circ *and an initialized dictionary* Γ_i*. Then concurrently execute the following for each linear, multiplication, or output gate:*

LINEAR GATE $(G, \mathsf{linear}, a_0, a_1, G_1, \ldots, a_l, G_l)$ (only for $i \neq k$):

1. wait until $\Gamma_i(k, G_u) \neq \perp$, for all $u = 1 \ldots l$.
2. compute $\Gamma_i(k, G) := A_0 \oplus \left(\bigoplus_{u=1}^{l} (a_u \cdot \Gamma_i(k, G_u)) \right)$.

MULTIPLICATION GATE $(G, \mathsf{mul}, G_1, G_2)$:

1. wait until $\Gamma_i(k, G_1) = C_1 \neq \perp$ and $\Gamma_i(k, G_2) = C_2 \neq \perp$
2. help to generate a randomizer (R, U, cert) for G:
   (a) compute $R_i = \mathscr{E}(r_i)$ and $U_i = \mathscr{R}(r_i C_1)$ for a randomly picked $r_i \in \mathcal{M}$
   compute $\sigma_i := \mathsf{Sign}_i(\text{«}(R_i, U_i) : \textit{part of } P_k\textit{'s randomizer for } G\text{»})$
   construct a proof
   $\pi_i := \mathsf{proof}_i(\text{«}P_i \textit{ knows } r_i \textit{ in } R_i, \textit{ and } U_i \textit{ is a randomization of } r_i C_1\text{»})$
   (b) send $(R_i, U_i, \sigma_i, \pi_i)$ to king P_k
   (c) wait until received from P_k set
   $S_G := \{(R_l, U_l, \sigma_l, \pi_l)\}_{l \in I_G}$, with $|I_G| \geq t + 1$
   (d) compute $R := \bigoplus_{i \in I_G} R_i$, and $U := \bigoplus_{i \in I_G} U_i$
   compute $\mathsf{cert}_i = \mathscr{S}_i(\text{«}(R, U) : P_k\textit{'s randomizer for } G\text{»})$
   (e) send cert_i to king P_k
3. wait until received (R, U, cert) from P_k,
   with $\mathscr{V}(\text{«}(R, U) : P_k\textit{'s randomizer for } G\text{»}, \mathsf{cert}) = 1$
4. compute z_i, P_i's decryption share for $Z = C_2 \oplus R$, and $\phi_i = \mathsf{proof}(\text{«}z_i \textit{ is valid}\text{»})$;
   send (z_i, ϕ_i) to P_k
5. wait until received V_G from P_k, with $|V_G| \geq t + 1$
6. decrypt $z := \mathscr{D}(Z, V_G)$ and compute $\Gamma_i(k, G) := (z \cdot C_1) \ominus U$

OUTPUT GATE (G, output, G'):

1. wait until $\Gamma_i(k, G') = C \neq \perp$
2. compute a decryption share $c_i := \mathscr{D}_i(C)$ and a proof $\varpi_i = \mathsf{proof}(\text{«}c_i \textit{ is valid}\text{»})$;
   send (c_i, ϖ_i) to P_k

Fig. 3. The code for slave P_i helping king P_k to evaluate the circuit

the encryption, and the fact that $c_1 c_2 = (c_2 + r)c_1 - rc_1$. That is, P_i computes $\Gamma_i(k, G) := (z \cdot C_1) \ominus U$.

3.5 Output Stage

When P_i notices that the computation of an output gate (G, output, G') is completed by some king P_k (i.e. $\Gamma_i(k, G) = C \neq \perp$), but the gate has not been decrypted so far, then P_i sends a decryption share c_i of C to P_k along with a proof that the decryption share is correct. Then P_k collects enough decryption shares, and computes the value of the output gate.

During the protocol each party executes concurrently the following rules:

RULE 1:
1. wait until the output gate $G \in G(\mathcal{C})$ is marked decrypted
2. vote by sending the value of the gate to all parties

RULE 2:
1. wait until receiving $t+1$ identical votes for the value of the output gate
2. adopt the value receiving $t+1$ votes
3. mark the output gate $G \in G(\mathcal{C})$ as decrypted

RULE 3:
1. wait until receiving $n-t$ identical votes for the value of the output gate
2. terminate

Fig. 4. The code for terminating P_i

3.6 Termination

As described above each king P_k will eventually learn the value of the output gate. This however requires that each slave P_i keeps running after king P_k learned the output values. To allow to also terminate the slaves, the parties execute a termination protocol. When king P_k learns the output of the circuit it outputs it and echos the result to all parties as its *vote* for the output (and does not yet terminate slave P_k). Since all honest parties compute identical outputs and there are at most t corrupted parties, if a party receives $t+1$ identical votes for some output value it can safely adopt this value as its own output, terminate its own king, and then echo the adopted output value. When a party receives $(n-t)$ identical votes for the output value it terminates the protocol. This is essentially a Bracha broadcast of the output value and allows all parties to eventually terminate.

3.7 Security Analysis

Our protocol can be proved secure in the model described in Section 2. A formal proof that the protocol can be simulated can be given along the lines of the proof in [CDN01], using the following helping lemmas.

Lemma 1 (The correctness invariant). *The Properties 1, 2, 3 and 4 of Definition 1 hold at any point in the protocol if there are at most $t < n/3$ corrupted parties.*

Lemma 2 (Termination). *If all honest parties start running the protocol and there are at most $t < n/3$ corrupted parties, then all honest parties will eventually terminate the protocol.*

The proofs of the lemmas are given in the full version of the paper [HNP04]. Here we discuss how the lemmas allow to give a proof along the lines of [CDN01].

By property 1 a set of at least $n - t$ parties have their inputs considered, as required by the model in Section 2. Furthermore, by Property 3, the output v_i obtained by P_i when decrypting the output ciphertext in Step 3 in OUTPUT GATE in Fig. 2 will be correctly defined from the plaintexts of the input ciphertexts held by P_i. Since by Property 2 the parties agree on the input ciphertexts, all honest parties P_i will agree on the output v_i in Step 3 in OUTPUT GATE. This clearly implies that all honest parties terminate the protocol in Fig. 4 with the output being the common value v, as no other value can get $t + 1$ votes when there are at most t corrupted parties. Since v is the result of evaluating the circuit on the plaintexts of the input ciphertexts and, by Property 2, the input ciphertext X_l of honest party P_l contains the correct input x_l, the result v can indeed be obtained by restricting the set of input providers to a set of size at least $n - t$ and then changing only the inputs of the corrupted parties.

The privacy of the protocol (formally defined by the simulator only being given the inputs of the corrupted parties in the simulation) mainly follows from the fact that all inputs are encrypted using a semantic secure encryption scheme and that all proofs are zero-knowledge. So, the only knowledge leaked about the inputs of the honest parties is through decryptions of ciphertexts.

The decryptions take place only in Step 4 in MULTIPLICATION in Fig. 3 and in Step 2 in OUTPUT GATE in Fig. 3. By the correctness of the protocol the knowledge leaked in Step 2 in OUTPUT GATE is the result of the computation, which is allowed to leak by the model. So it remains to argue that no knowledge is leaked in Step 4 in MULTIPLICATION. To see this, observe that the value revealed by the decryption in Step 4 is $z = c_2 + \sum_{i \in I_G} r_i$, which holds the potential of leaking knowledge about c_2 (which is potentially to be kept secret). Since each term r_i from an honest party is chosen uniformly at random and all r_i are chosen independently (this is the purpose of having all parties, in particular the corrupted parties, prove plaintext knowledge of their r_i in Step 2(a)), it is sufficient to show that each revealed value $z = c_2 + \sum_{i \in I_G} r_i$ contains at least one honest value r_i *which did not enter another revealed value.*

Observe first of all that since $|I_G| \geq t+1$, at least one r_i came from an honest party. Observe then that each of the randomizers r_i are associated uniquely to one (P_k, G) by the signature σ_i (issued in Step 2(a) and checked in Step 2(c)). Therefore r_i only enters values $z = c_2 + \sum_{i \in I_G} r_i$ leaked in decryptions in Step 4 in MULTIPLICATION for the specific (P_k, G) in consideration. It is therefore sufficient to show that for each (P_k, G) there is only one value $z = c_2 + \sum_{i \in I_G} r_i$ for which knowledge is leaked. This follows from the uniqueness guaranteed by the threshold $t_S = n - t$ of the threshold signatures. More precisely, assume that for each king P_k and each gate G at most one value of the form $\langle\!\langle \cdot : P_k$*'s randomizer for* $G \rangle\!\rangle$ is signed. I.e. there exists at most one value (R, U) for which there exists cert such that $\mathcal{V}(\langle\!\langle (R, U) : P_k$*'s randomizer for* $G \rangle\!\rangle, \mathsf{cert}) = 1$ (this can be seen to be necessary for Property 4 to hold and thus follows from Lemma 1). Since the honest parties agree on the gate encryptions of P_k (by Property 4), this implies that there is at most one value $Z = C_2 \oplus R$ for which honest parties issue decryption shares in Step 4 for a given choice of (P_k, G).

Therefore each value $z = c_2 + \sum_{i \in I_G} r_i$ on which knowledge is leaked through decryption shares from honest parties, at least one r_i came from an honest party and did not enter another value on which knowledge was leaked, as desired.

3.8 Efficiency Analysis

In this section we consider the communication complexity of the protocol. We omit computational complexity from the analysis, since it is clearly polynomial, and the bottleneck of distributed computing is in the communication overhead. For completeness, we consider the case where each party can have more than one input, and we denote by c_I the total number of input gates. For clarity we use $K = n$ to denote the number of kings and $S = n$ to denote the number of slaves.

In the protocol in Fig. 1, when each party has more than one input, X_i will simply be the vector of input encryptions. Assuming that all encryptions, all signature shares, all signatures and all pairwise proofs use communication $\mathcal{O}(\kappa)$ and that the communication complexity of a Byzantine agreement is $\mathcal{O}(n^2\kappa)$, it can be seen using simple counting that the communication complexity of the protocol in Fig. 1 is $\mathcal{O}(c_I n^2 \kappa + n^3 \kappa)$.

In king's protocol (Fig. 2) the only values sent are the sets S_G and V_G. These sets have size $\mathcal{O}(n\kappa)$ and are sent to all S slaves. This gives a communication complexity of $\mathcal{O}(Sn\kappa)$ each time a set is sent, or a total communication complexity of $\mathcal{O}((c_M + c_O)Sn\kappa)$ for running the protocol in Fig. 2, where c_M is the number of multiplication gates and c_O is the number of output gates. This is done by all K kings, yielding a total communication complexity of $\mathcal{O}((c_M + c_O)KSn\kappa)$.

In slave's protocol (Fig. 3) the sending of the values in Steps 2(b), 2(e) and 4 of MULTIPLICATION GATE, and Step 2 of OUTPUT GATE all use $\mathcal{O}(\kappa)$ bits of communication. Moreover, the constructions of the proofs in Steps 2(a) and 4 of MULTIPLICATION GATE, and in Step 2 of OUTPUT GATE takes $\mathcal{O}(n\kappa)$ bits of communication, and thus are the dominating instructions. Each construction of a proof is done at most once for each gate for each king being helped. This yields a total of $\mathcal{O}((c_M + c_O)Kn\kappa)$ for running the slave's protocol. Since the protocol is run by all S slaves, this yields a total communication complexity of $\mathcal{O}((c_M + c_O)KSn\kappa)$.

It is easy to verify that the total communication complexity of the protocol in Fig. 4 is $\mathcal{O}(c_O n^2 \kappa)$.

Summing the above terms we get a communication complexity of $\mathcal{O}((c_I + c_O)n^2\kappa + (c_M + c_O)KSn\kappa + n^3\kappa)$. Using $K = S = n$ and assuming that $c_O \geq 1$, this is $\mathcal{O}(c_I n^2 \kappa + (c_M + c_O)n^3\kappa)$, as claimed.

4 Extensions and Applications

4.1 Computing Functions with Private Outputs

The description of the new protocol in Section 3 only considers public outputs, i.e., every party learns the output(s) of the circuit. In the following, we present an extension that allows for outputs that are delivered only to an authorized party, say P_j.

The intuition of the protocol is that the decryption shares are not sent to the king, but rather directly to P_j. Every decryption share must go along with a proof of validity. This proof must not be interactive with P_j (the parties cannot wait for messages of P_j), and the proof must not be given to other parties (this would violate the privacy of the output protocol). Therefore, we have every slave P_i blind his decryption share c_i with a random value r_i, i.e., $c'_i = c_i + r_i$, encrypt r_i with randomness ρ_i, i.e., $R_i = \mathscr{E}(r_i, \rho_i)$, and prove interactively towards every auxiliary player P_j knowledge of r_i such that r_i encrypts to R_i and $c'_i - r_i$ is a valid decryption share. Upon accepting the proof, every auxiliary player hands a signature share for «(c'_i, R_i) *is a good decryption share for slave* P_i» to P_i, who then sends c'_i, r_i, ρ_i to P_j. Given this information from at least $n - t$ players, P_j picks the valid decryption shares and decrypts his private output.

We note that a similar technique has been recently used by Schoenmakers and Tuyls [ST04].

4.2 A Hybrid Model: Asynchronous Network with Few Synchronization Rounds

A fully asynchronous MPC protocol inherently cannot consider the input of every honest party; once $n - t$ inputs are ready, the protocol must start. This is a serious drawback which makes the fully asynchronous model unusable for many real-world applications. We show that with a single round of synchronization, we can consider the input of *every* honest party. This model seems very reasonable in real-world; the parties would wait for other parties to have their input ready, and if not, use other means of communication (email, phone, fax, etc) to synchronize. However, the MPC protocol itself should run asynchronously to comply with the properties of existing networks, namely that the delay of messages is hard to predict. Note that asynchronous protocols can be looked as "best effort" protocols where the progress in the protocol is as fast as possible with the available network, in contrast to synchronous protocols whose progress is limited by the assumed worst-case delay of the network.

The necessary changes in the input protocol (cf. Fig. 1) are minimal: Every player P_i moves to the last stage (Select Input Providers) only when either $|C_i| = n$, or the synchronization round elapsed.

4.3 Non-robust Computations

In the proposed protocol, robustness is guaranteed by having each party act as king, who evaluates the whole circuit on his own (with help of his slaves). This means that the computation and communication overhead for achieving robustness is a factor of n.

Goldwasser and Lindell have proposed a model for secure MPC in which output delivery is not guaranteed [GL02], unless some a priori specified party is honest. In this model, we can improve the communication complexity of our protocol by a factor of n, simply be letting this designated party act as king, and all other parties act as his slaves. We stress that the protocol still guarantees

privacy and correctness of the computation, but termination (with output delivery) can only be guaranteed when the king is honest. This simplified protocol achieves an overall communication complexity of $\mathcal{O}(cn^2\kappa)$ for a circuit of size c and a security parameter κ.

5 Conclusions

We have proposed a secure multi-party computation protocol which substantially puts forward both theory and practice in this field. From a theoretical point of view, the protocol is optimally resilient, fully asynchronous, and has an asymptotically lower communication complexity than any previous asynchronous MPC protocol. Indeed, the protocol is as efficient as the most efficient known protocol for synchronous communication. Furthermore, the protocol requires very few invocations of the broadcast primitive (independent of the size of the computed circuit).

From a practical point of view, the new protocol is designed for real-world networks with unknown message delay, allows every party to provide his input under a very reasonable assumption (one round of synchronization), and achieves best-possible resilience against cheating (up to a third of the parties may misbehave). Furthermore, the protocol is very efficient, the constant communication overhead is minimal. The effective computation of the circuit takes less than $10n^3\kappa$ bits of communication per multiplication, which makes the protocol applicable for reasonably sized circuits among small sets of parties. The key set-up (for the threshold decryption and threshold signatures) is more communication-intensive; however, this can be performed long in advance.

Acknowledgements. We would like to thank anonymous referees for helpful comments.

References

[BCG93] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proc. 25th ACM STOC*, pages 52–61, 1993.

[Bec54] Samuel Beckett. *Waiting for Godot*. New York: Grove Press, 1954.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th ACM STOC*, pages 1–10, 1988.

[BKR94] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. Asynchronous secure computations with optimal resilience. In *Proc. 13th ACM PODC*, pages 183–192, 1994.

[Can95] Ran Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE FOCS*, pages 136–145, 2001.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. 20th ACM STOC*, pages 11–19, 1988.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. EUROCRYPT '01*, pages 280–300, 2001.

[CKS00] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in Constantinopole: Practical asynchronous Byzantine agreement using cryptography. In *Proc. 19th ACM PODC*, pages 123–132, 2000.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proc. 4th PKC*, pages 110–136, 2001.

[FH96] Matt Franklin and Stuart Haber. Joint encryption and message-efficient secure computation. *Journal of Cryptology*, 9(4):217–232, 1996.

[FPS00] Pierre-Alain Fouque, Gouillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Proc. Financial Cryptography '00*, 2000.

[GL02] S. Goldwasser and Y. Lindell. Secure computation without agreement. In *DISC '02*, pages 17–32, 2002.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game — a completeness theorem for protocols with honest majority. In *Proc. 19th ACM STOC*, pages 218–229, 1987.

[HNP04] Martin Hirt, Jesper Buus Nielsen, and Bartosz Przydatek. Cryptographic asynchronous multi-party computation with optimal resilience, 2004. *Cryptology ePrint Archive*, eprint.iacr.org, report no. 2004/368.

[Nie02] Jesper B. Nielsen. A threshold pseudorandom function construction and its applications. In *Proc. CRYPTO '02*, pages 401–416, 2002.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. EUROCRYPT '99*, pages 223–238, 1999.

[PSR02] B. Prabhu, K. Srinathan, and C. Pandu Rangan. Asynchronous unconditionally secure computation: An efficiency improvement. In *Proc. Indocrypt 2002*, pages 93–107, 2002.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sho00] Victor Shoup. Practical threshold signatures. In *Proc. EUROCRYPT '00*, pages 207–220, 2000.

[SR00] K. Srinathan and C. Pandu Rangan. Efficient asynchronous secure multiparty distributed computation. In *Proc. Indocrypt 2000*, pages 117–129, 2000.

[ST04] Berry Schoenmakers and Pim Tuyls. Practical two-party computation based on the conditional gate. In *Proc. ASIACRYPT '04*, pages 119–136, 2004.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *Proc. 23rd IEEE FOCS*, pages 160–164, 1982.

Differential Cryptanalysis for Multivariate Schemes

Pierre-Alain Fouque, Louis Granboulan, and Jacques Stern

École normale supérieure,
Département d'Informatique 45, rue d'Ulm,
75230 Paris cedex 05, France
{Pierre-Alain.Fouque, Louis.Granboulan, Jacques.Stern}@ens.fr

Abstract. In this paper we propose a novel cryptanalytic method against multivariate schemes, which adapts differential cryptanalysis to this setting. In multivariate quadratic systems, the differential of the public key is a linear map and has invariants such as the dimension of the kernel. Using linear algebra, the study of this invariant can be used to gain information on the secret key. We successfully apply this new method to break the original Matsumoto-Imai cryptosystem using properties of the differential, thus providing an alternative attack against this scheme besides the attack devised by Patarin. Next, we present an attack against a randomised variant of the Matsumoto-Imai cryptosystem, called PMI. This scheme has recently been proposed by Ding, and according to the author, it resists all previously known attacks. We believe that differential cryptanalysis is a general and powerful method that can give additional insight on most multivariate schemes proposed so far.

1 Introduction

The design of efficient and secure cryptosystems is a hard task. Many alternatives to the traditional public key cryptosystems (RSA, ElGamal) have been proposed so far but few of them are considered secure. An interesting line of research is based on multivariate quadratic polynomials over a finite field. This line of research has been initiated by Matsumoto and Imai [12]. These systems are attractive since the underlying problem is known to be NP-complete and the decryption algorithm is more efficient than the RSA algorithm.

The original cryptosystem of Matsumoto and Imai (MI or C^*) has been broken by Patarin [13] who has also proposed various techniques that protect against this attack [15, 14]. A generalisation of MI, called Hidden Field Equations (HFE) [17], has higher security, but it has nevertheless been broken by Kipnis and Shamir [11]. More efficient attacks were proposed by Courtois *et al.* in [5, 6] and culminated with Faugère and Joux attack and the use of Gröbner bases in [9].

Variants of the original MI scheme remain interesting because they achieve better performance than variants of HFE. The main variants of MI that resist the attack by Patarin are on one hand, the Minus method which consists

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 341–353, 2005.

in discarding a few polynomials in the public key, and on the other hand the Minus-Plus method, which proposes to discard some polynomials and to add a few variables. These methods use *external* perturbation of the MI scheme, since variables are removed after the application of the exponentiation function.

Recently, Ding [7] proposed a new variant of the MI cryptosystem using some *internal* perturbation, which occurs before applying the exponentiation function. He quickly analyses its proposal against all known attacks on multivariate schemes, and claims that it is immune against such attacks. The new scheme is nearly as efficient as the original MI and the author gives some arguments in order to show that its scheme, called Perturbated MI (PMI), is a more secure extension than the MI Minus and MI Minus-Plus method.

1.1 Our Results

In this paper, we describe a new technique which is extremely powerful and that could presumably be used to break other multivariate schemes. In order to illustrate the power and generality of this method, we first propose a new attack on the original MI scheme and next describe how it can be used to mount an attack against the PMI cryptosystem.

The key point of our attack is that in the case of quadratic polynomials, the differential of the public key is a linear map and its kernel or its rank can be analysed to get some information on the secret key. For example, in the PMI scheme, we show that the dimension of the kernel can be used to identify elements that cancel the perturbation. In fact, we design a one-sided error recogniser for the language of elements that are not in the kernel of the perturbation. From this test algorithm, we design two algorithms to reconstruct the kernel. These algorithms are of independent interest. With the first method, the complexity of the attack is a precomputation of order $O(nq^{3r} + n^6 q^r)$, which can be upperbounded by 2^{49} with the proposed parameters in [7], and $O(n^3 \times q^r \times q^{\gcd(\ell,n)})$, which is of order 2^{36} binary operations. Finally, this attack works for scheme over finite fields of characteristic 2 which are the main structure for efficiency reasons and for MI and PMI this is always the case as we will see. In the case of the original MI cryptosystem, we use elements in the kernel of the transpose of the differential in order to propose a new attack. We actually prove a bilinear relation between the ciphertext and the kernel vector. Thus, the kernel allows to recover the plaintext by solving a linear system.

1.2 Related Works

Differentials have already been successfully applied to break multivariate schemes such as the Minus transformation of the original Matsumoto-Imai, or the SFLASH signature scheme or the "2R" scheme proposed by Patarin [14, 16, 10, 8]. Our work gives a better insight by bringing a systematic use of the geometric properties of the differential.

1.3 Organisation of the Paper

In section 2 of this paper, we describe the MI and PMI cryptosystems. Then, in section 3 we recall Patarin's attack on the original MI scheme. Next in section 4, we describe our attack on the PMI scheme and some experimental results. Finally, in section 5, we show a new attack on the original MI scheme.

2 Description of the MI and PMI Schemes

2.1 The Matsumoto-Imai Cryptosystem

This scheme is based on the following fact : over the finite field $\mathbb{F}_{q^n}$, the function $F : x \mapsto x^{q^\ell+1}$ is a permutation, when $\gcd(q^\ell+1, q^n-1) = 1$. Therefore, we can fix q to be a power of 2 so that $\mathbb{F}_{q^n}$ is of characteristic two [1]. Its inverse is $x \mapsto x^h$ where h is the inverse of $q^\ell+1$ in $\mathbb{Z}_{q^n-1}$. Therefore, for any isomorphism π from the vector space of $\mathbb{F}_{q^n}$ to the n-dimensional vector space $(\mathbb{F}_q)^n$, the function $\boldsymbol{F} = \pi \circ F \circ \pi^{-1}$ is a bijective system of multivariate quadratic functions since F can be viewed as the product of two linear maps $x \mapsto x^{q^\ell}$ and $x \mapsto x$.

The scheme described by Matsumoto and Imai in 1988 [12] generates $\boldsymbol{S}$ and $\boldsymbol{T}$, two secret affine bijections of $(\mathbb{F}_q)^n$ to mask the system $\boldsymbol{F}$. The system $\boldsymbol{E} = \boldsymbol{T} \circ \boldsymbol{F} \circ \boldsymbol{S}$ is also a system of multivariate quadratic equations and represents the public key. Patarin showed in 1995 [13] that the public key has a special form which allows to invert the function.

2.2 The PMI Cryptosystem

Recently at PKC '04, Ding proposed a randomised variant of MI, called PMI [7]. Let $\boldsymbol{R} : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^r$ a secret linear function of small rank ($r \ll n$) and $\boldsymbol{H}$ a secret quadratic system composed of n quadratic equations over r variables. The PMI public key is the system $\boldsymbol{E'}$ defined by $\boldsymbol{E'} = \boldsymbol{T} \circ (\boldsymbol{F} + \boldsymbol{H} \circ \boldsymbol{R}) \circ \boldsymbol{S}$. The public key can also be written as $\boldsymbol{E'} = \boldsymbol{T} \circ \boldsymbol{F} \circ \boldsymbol{S} + \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}$ due to the linearity of $\boldsymbol{T}$. Consequently, the PMI scheme can be seen as the MI scheme $\boldsymbol{E}$ plus a random-looking quadratic term $\boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}$. Since there is no trapdoor to invert $\boldsymbol{H}$ or to separate the MI term and the random term, we need to store all the inputs and the outputs of the $\boldsymbol{H}$ function. Let P be the set of points which consist of pairs $(\boldsymbol{\lambda}, \boldsymbol{\mu})$, where $\boldsymbol{\lambda}$ is a point that belongs to the image of $\boldsymbol{H}$, and $\boldsymbol{\mu}$ is the set of pre-images of $\boldsymbol{\lambda}$ under $\boldsymbol{H}$. The set P contains q^r points. The secret key includes the set of linear functions $\boldsymbol{R}$, the set P, and the two affine bijections $\boldsymbol{S}$ and $\boldsymbol{T}$.

The secret key allows to invert $\boldsymbol{E'}$ if one can make exhaustive search over the q^r values of P and so r must be small. More precisely, given a ciphertext $\boldsymbol{y}$, the decryption process inverses the affine bijection $\boldsymbol{T}$ and recovers $\boldsymbol{y'}$. Then, all elements $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ in P can be tried one-by-one and $\boldsymbol{y'_\lambda} = \boldsymbol{F}^{-1}(\boldsymbol{y'} + \boldsymbol{\lambda})$ is computed.

[1] Indeed, if q is odd, we have $\gcd(q^\ell+1, q^n-1) \geq 2$ and since q is a prime power, it is always a power of 2 and the characteristic of $\mathbb{F}_{q^n}$ is 2.

Next, if $\boldsymbol{H}(\boldsymbol{y}'_{\boldsymbol{\lambda}})$ is not equal to $\boldsymbol{\mu}$, we try the next point in P, otherwise, we compute $\boldsymbol{x}_{\boldsymbol{\lambda}}$ by $\boldsymbol{S}^{-1}(\boldsymbol{y}'_{\boldsymbol{\lambda}})$. If we have only one solution, we get the plaintext, otherwise, we use some added redundancy in the plaintext in order to uniquely recover it.

In his description of PMI [7], Ding analyses all known attack such as algebraic attacks of Patarin [13], Kipnis and Shamir [11], or XL attacks [4] and the attack on MI Minus of Patarin, Goubin and Courtois [16].

He also proposes a practical implementation with $q = 2$, $n = 136$, $r = 6$ and $F(x) = x^{2^{5\times 8}+1}$. He claims that the security level for this choice of parameters is 2^{136}. The value ℓ has been chosen with a special form, such that $\gcd(2^n - 1, 2^\ell - 1) = 2^{\gcd(n,\ell)} - 1 = 2^{\gcd(136,5\times 8)} - 1 = 2^8 - 1$. This special form allows to perform more efficient encryption and decryption using lookup tables for the multiplications in the finite field.

In this paper, we apply differential cryptanalysis to the PMI scheme, and we show that the special form of the exponent in the practical system proposed by Ding allows more efficient attack than the attack in the generic case where $\gcd(\ell, n) = 1$.

3 Patarin's Attack on the MI Cryptosystem

Our attack against the PMI cryptosystem is a probabilistic reduction to Patarin's attack on the MI scheme. Therefore, prior the description of our attack, we recall Patarin's attack. While we also propose an alternative attack to the MI scheme in section 5, we present Patarin attack since it is easier to understand. Both his attack and our attack do not recover the secret key but finds a linear system which can be solved to recover the plaintext corresponding to a given ciphertext.

Let $\boldsymbol{x} \in (\mathbb{F}_q)^n$ a plaintext and $\boldsymbol{y} \in (\mathbb{F}_q)^n$ the corresponding ciphertext. The main idea of Patarin attack is to find several bilinear relations in the $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates. Using plaintext/ciphertext pairs $(\boldsymbol{x}, \boldsymbol{y})$, it is possible to recover the coefficients of the relations by solving a linear system. Finally, knowing these coefficients and a given ciphertext, it is possible to decrypt $\boldsymbol{y}$ by solving a linear system.

Let us define $a = \pi^{-1}(\boldsymbol{S}(\boldsymbol{x}))$ and $b = \pi^{-1}(\boldsymbol{T}^{-1}(\boldsymbol{y}))$. Consequently, $F(a) = b$ or $b = a^{q^\ell+1}$. By raising each member of the last equation to the power $q^\ell - 1$ and by multiplying each one by ab, we get

$$ab^{q^\ell} = a^{q^{2\ell}} b \tag{1}$$

which holds over the finite field $\mathbb{F}_{q^n}$. We can rewrite this equation by $B(a, b) = 0$ where $B(a, b) = a \cdot b^{q^\ell} - a^{q^{2\ell}} \cdot b$. If we represent equation (1) in $(\mathbb{F}_q)^n$, we get n bilinear equations in the n coordinates of $\boldsymbol{a}$ and of $\boldsymbol{b}$. As $\boldsymbol{a}$ and $\boldsymbol{b}$ are affine transformations of $\boldsymbol{x}$ and $\boldsymbol{y}$ via the secret affine bijections $\boldsymbol{S}$ and $\boldsymbol{T}$, the n bilinear expressions in $\boldsymbol{a}$ and $\boldsymbol{b}$, may also be written as n bilinear expressions in $\boldsymbol{x}$ and $\boldsymbol{y}$. Each expression can be written as $\sum_{i=1}^n \sum_{j=1}^n \beta_{i,j} x_i y_j + \sum_{i=1}^n \beta_{i,0} x_i + \sum_{j=1}^n \beta_{0,j} y_j + \beta_{0,0} = 0$.

For each plaintext/ciphertext pair $(\boldsymbol{x}, \boldsymbol{y})$, the equation above, where all the $\beta_{i,j}$ are the $(n+1)^2$ unknowns, has at least the n solutions described by the n bilinear expressions deduced from equation (1). Therefore, using $\mathcal{O}((n+1)^2)$ plaintext/ciphertext pairs, solving the resulting system of $\mathcal{O}((n+1)^2)$ equations in the $(n+1)^2$ unknowns $\beta_{i,j}$ will recover the n bilinear expressions.

Finally, given a ciphertext $\boldsymbol{y}$ to decrypt, these n equations will give us n linear equations in the coefficients of $\boldsymbol{x}$. Unfortunately, all these equations are not independent. The solutions of this system correspond to the solutions of (1). There are $q^{\gcd(n,\ell)}$ such solutions, as shown by Patarin: let us consider the equation (1) where the unknown is a. A ciphertext $\boldsymbol{y}$ fixes a unique b value. One solution is $a = 0$. If $a \neq 0$ (and so $b \neq 0$) the equation can be written as

$$a^{q^{2\ell}-1} = b^{q^{\ell}-1}$$

We can write $q^{2\ell}-1$ as $(q^{\ell}+1)(q^{\ell}-1)$ and take the inverse of $q^{\ell}+1$ modulo q^n-1 since by assumption F is a permutation. Consequently, the equation becomes $a^{q^{\ell}-1} = b^{h\times(q^{\ell}-1)} = b'$ where h is the inverse of $q^{\ell}+1$. This last equation has exactly $\gcd(q^{\ell}-1, q^n-1) = q^{\gcd(\ell,n)}-1$ solutions as shown in appendix A since the right solution is one solution.

As a consequence, the solution that we are looking for is a particular vector of the kernel of some system related to the original system, and the second member of the equation [3–p. 59]. Therefore, we compute the kernel of the system matrix which is of dimension $\gcd(n, \ell)$. Next, we perform an exhaustive search in $q^{\gcd(n,\ell)} - 1$ coefficients of the kernel vector, in order to recover the correct value $\boldsymbol{x}$.

In section 5, we propose a new differential attack on the MI scheme by studying the kernel of the transpose of the differential of the public key. We show that there exist n bilinear forms between a ciphertext $\boldsymbol{E}(\boldsymbol{k})$ and the vector $\boldsymbol{f_k}$ that generates the kernel of the transpose of the differential, which is of dimension 1 if $\gcd(\ell, n) = 1$. Then, given a ciphertext, we are able to reconstruct the vector $\boldsymbol{f_k}$ since the n bilinear forms are independent as there is a unique solution for the n bilinear forms. Finally, since the vector $\boldsymbol{f_k}$ is in the kernel of the transpose of the differential and that this map is linear in $\boldsymbol{k}$, we can solve n linear equations in the $\boldsymbol{k}$ variables of n coordinates. We refer the reader to section 5 for details.

4 Cryptanalysis of the PMI Cryptosystem

4.1 Overview of the Attack

Let us recall the notations : $\boldsymbol{F}$ is the system of quadratic equations corresponding to the internal function of the MI cryptosystem, $\boldsymbol{E} = \boldsymbol{T} \circ \boldsymbol{F} \circ \boldsymbol{S}$ the public key of MI, and $\boldsymbol{E'} = \boldsymbol{E} + \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}$ the public key of PMI.

Our attack is based on the following remark: the PMI scheme is a noisy MI cryptosystem. We find the linear space $\mathcal{K}$ that cancels the noise, and apply an attack of MI to the restriction of PMI to this linear space.

More precisely, we define the linear space $\mathcal{K}$ as follows: it is the kernel of the linear part of the affine function $\boldsymbol{R} \circ \boldsymbol{S}$. The space $\mathcal{K}$ is of dimension $\dim(\ker \boldsymbol{R}) = n - r$ because $\boldsymbol{S}$ is a bijection and $\operatorname{rank}(\boldsymbol{R}) = r$. If we are able to compute $\mathcal{K}$, then we can apply the attacks against MI (either Patarin's attack or our attack described in section 5) to the PMI cryptosystem restrict to elements of one of the q^r affine spaces that are parallel to $\mathcal{K}$. When restrict to one of these affine spaces, the public key of PMI is exactly $\boldsymbol{E}$ translated by a constant. The attack of PMI amounts to q^r attacks against MI (this is feasible because q^r must be of moderate size to allow fast decryption). A ciphertext is decrypted by applying the attack to the affine space that contains its corresponding plaintext.

In order to recover the space $\mathcal{K}$, we devise an efficient test algorithm that can spot that a given vector $\boldsymbol{k}$ does not belong to $\mathcal{K}$. The information used in this test is the dimension of the kernel of the linear part of the differential of the public key.

4.2 The Dimension of the Kernel of the Differential

For any function $\boldsymbol{G} : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^m$, let us consider its differential $\boldsymbol{dG_k}(x) = \boldsymbol{G}(\boldsymbol{x} + \boldsymbol{k}) - \boldsymbol{G}(\boldsymbol{x})$. Because $\boldsymbol{G}$ is a quadratic function, its differential is an affine function. Let us consider $\boldsymbol{L_{G,k}}(x) = \boldsymbol{dG_k}(x) - \boldsymbol{dG_k}(0)$ the linear part of the differential. In fact, it is a bilinear function that can also be defined by $\boldsymbol{L_{G,k}}(x) = \boldsymbol{B_G}(\boldsymbol{x}, \boldsymbol{k}) = \boldsymbol{G}(\boldsymbol{x}+\boldsymbol{k}) - \boldsymbol{G}(\boldsymbol{x}) - \boldsymbol{G}(\boldsymbol{k}) + \boldsymbol{G}(\boldsymbol{0})$, and is also called the polar form. We are interested in $\dim(\ker \boldsymbol{L_{G,k}})$ when $\boldsymbol{G}$ is the public key of the cryptosystem.

Property 1. Let $\boldsymbol{k}$ and $\boldsymbol{k'}$ be elements of $(\mathbb{F}_q)^n$, and $\boldsymbol{G}$ and $\boldsymbol{G'}$ be systems of quadratic equations, and $\boldsymbol{S}$ and $\boldsymbol{T}$ be affine bijections. The following properties hold: $\boldsymbol{L_{G,k+k'}} = \boldsymbol{L_{G,k}} + \boldsymbol{L_{G,k'}}$, $\boldsymbol{L_{G+G',k}} = \boldsymbol{L_{G,k}} + \boldsymbol{L_{G',k}}$, $\boldsymbol{L_{T \circ G \circ S,k}} = \boldsymbol{T} \circ \boldsymbol{L_{G,S(k)}} \circ \boldsymbol{S} + \boldsymbol{T} \circ \boldsymbol{G} \circ \boldsymbol{S}(\boldsymbol{0}) - \boldsymbol{T} \circ \boldsymbol{G}(\boldsymbol{0})$, and $\boldsymbol{L_{G,0}} = \boldsymbol{0}$.

Lemma 1. *If $\boldsymbol{E}$ is the public key of a MI system over $\mathbb{F}_q$ of characteristic 2, of dimension n and exponent $q^\ell + 1$, then* $\dim(\ker \boldsymbol{L_{E,k}}) = \gcd(\ell, n)$.

First, $\dim \ker(\boldsymbol{L_{E,k}}) = \dim \ker(\boldsymbol{L_{F,k}})$, because $\boldsymbol{T}$ and $\boldsymbol{S}$ are bijections.

Let us define $\boldsymbol{x} = \pi(x)$ and $\boldsymbol{k} = \pi(k)$. If $\boldsymbol{F}$ is the internal function of the MI cryptosystem, then $\boldsymbol{B_F}(\boldsymbol{x}, \boldsymbol{k})$ is equal to $\pi(x^{q^\ell} \cdot k + x \cdot k^{q^\ell})$. A vector $\boldsymbol{x} \neq \boldsymbol{0}$ of $(\mathbb{F}_q)^n$ is in the kernel of $\boldsymbol{L_{F,k}}$ if and only if $x^{q^\ell} \cdot k + x \cdot k^{q^\ell} = 0$. This last equation can be written as $x^{q^\ell+1} \cdot \left(\frac{k}{x} + \left(\frac{k}{x}\right)^{q^\ell} \right) = 0$.

Since $x \neq 0$, if we denote k/x by X, then the previous equation is $X + X^{q^\ell} = 0$ in the finite field $\mathbb{F}_{q^n}$. If $X \neq 0$ ($k \neq 0$), then the equation becomes $X^{q^\ell - 1} = 1$ in a finite field of characteristic 2. Since $X = 1$ is solution, there is at least one solution. As a consequence, there are $q^{\gcd(\ell,n)} - 1$ solutions according to the results in appendix A, and therefore $\dim(\ker \boldsymbol{L_{E,k}}) = \gcd(\ell, n)$.

Note that $X = 1$ is always a solution, that means $x = k$, and therefore $\boldsymbol{k}$ is always in the kernel. There are no other solutions when $\gcd(\ell, n) = 1$.

Lemma 2. *If* $\boldsymbol{E'}$ *is the public key of the PMI cryptosystem and* $\boldsymbol{k} \in \mathcal{K}$*, then* $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) = \gcd(\ell, n)$.

We prove that if $\boldsymbol{k} \in \mathcal{K}$, then $\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}} = \boldsymbol{L}_{\boldsymbol{E},\boldsymbol{k}}$. First we notice that $\boldsymbol{k} \in \mathcal{K}$ is equivalent to $\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{k}) = \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{0})$.

Then we compute $\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}(\boldsymbol{x}) - \boldsymbol{L}_{\boldsymbol{E},\boldsymbol{k}}(\boldsymbol{x}) = \boldsymbol{L}_{\boldsymbol{T}\circ\boldsymbol{H}\circ\boldsymbol{R}\circ\boldsymbol{S},\boldsymbol{k}}(\boldsymbol{x}) = \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{x} + \boldsymbol{k}) - \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{x}) - \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{k}) + \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{0})$, therefore $\boldsymbol{T}^{-1}(\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}(\boldsymbol{x}) - \boldsymbol{L}_{\boldsymbol{E},\boldsymbol{k}}(\boldsymbol{x})) = \boldsymbol{H}(\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{x} + \boldsymbol{k})) - \boldsymbol{H}(\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{x})) - \boldsymbol{H}(\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{k})) + \boldsymbol{H}(\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{0})) = \boldsymbol{0}$, which means that $\boldsymbol{T}^{-1} \circ \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}} = \boldsymbol{T}^{-1} \circ \boldsymbol{L}_{\boldsymbol{E},\boldsymbol{k}}$. Therefore $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) = \dim(\ker(\boldsymbol{T}^{-1} \circ \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}})) = \dim(\ker(\boldsymbol{T}^{-1} \circ \boldsymbol{L}_{\boldsymbol{E},\boldsymbol{k}})) = \dim(\ker \boldsymbol{L}_{\boldsymbol{E},\boldsymbol{k}})$.

Lemma 3. *If* $\boldsymbol{E'}$ *is the public key of the PMI cryptosystem and* $\boldsymbol{k} \notin \mathcal{K}$*, then often* $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) \neq \gcd(\ell, n)$.

As before, $\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}$ is the sum of $\boldsymbol{L}_{\boldsymbol{E},\boldsymbol{k}}$ and $\boldsymbol{L}_{\boldsymbol{T}\circ\boldsymbol{H}\circ\boldsymbol{R}\circ\boldsymbol{S},\boldsymbol{k}}$. However, when, $\boldsymbol{k} \notin \mathcal{K}$, the second linear application is not null. The argument behind lemma 3 is that $\boldsymbol{L}_{\boldsymbol{T}\circ\boldsymbol{H}\circ\boldsymbol{R}\circ\boldsymbol{S},\boldsymbol{k}}$ is a random-looking linear application, and therefore the dimension of the kernel of the sum $\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}$ follows the distribution of the dimension of the kernel of random linear maps.

In fact, it is slightly more complicated, because $\boldsymbol{k}$ is always in the kernel of $\boldsymbol{L}_{\boldsymbol{T}\circ\boldsymbol{H}\circ\boldsymbol{R}\circ\boldsymbol{S},\boldsymbol{k}}$, and therefore also in the kernel of $\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}$, whose dimension then is at least 1. Moreover, if $\gcd(\ell, n) > r$, then there are $\gcd(\ell, n) - r$ additional vectors in the kernel of $\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}$, because $\ker(\boldsymbol{L}_{\boldsymbol{E},\boldsymbol{k}})$ of dimension $\gcd(\ell, n)$ and $\ker(\boldsymbol{R} \circ \boldsymbol{S})$ of dimension $n - r$ in a space of dimension n have an intersection of dimension at least $\gcd(\ell, n) - r$. In the case of the practical scheme proposed by Ding where $\gcd(\ell, n) = 8$ and $r = 6$, we can deduce that $\dim(\ker(\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}})) \geq 3$.

Lemma 3 can be verified experimentally, as shown in table 1.

Table 1. Experimental results for the probability distribution of $\dim(\ker(\boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}))$

$\ell = 41$, $n = 137$ and $r = 6$

dimension	$\boldsymbol{k} \in \mathcal{K}$	$\boldsymbol{k} \notin \mathcal{K}$
1	1	≈ 0.59
> 1	0	≈ 0.41

$\ell = 40$, $n = 136$ and $r = 6$

dimension	$\boldsymbol{k} \in \mathcal{K}$	$\boldsymbol{k} \notin \mathcal{K}$
3	0	≈ 0.686
4	0	≈ 0.290
5	0	≈ 0.023
6	0	$\approx 5.10^{-4}$
7	0	$\approx 2.10^{-6}$
8	1	≈ 0
> 8	0	≈ 0

As a consequence of the lemmas, we get the following corollary.

Corollary 1. *If* $\boldsymbol{E'}$ *is the public key of the PMI cryptosystem and if* $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) \neq \gcd(\ell, n)$*, then* $\boldsymbol{k} \notin \mathcal{K}$.

In conclusion, we have now an efficient test to know if a vector is not in $\mathcal{K}$. We define $T(\boldsymbol{k})$ to be this test: $T(\boldsymbol{k}) = 1$ if $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) \neq \gcd(\ell, n)$, meaning that

$\boldsymbol{k}$ is not in $\mathcal{K}$ with probability one, and $T(\boldsymbol{k}) = 0$ if $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) = \gcd(\ell, n)$, meaning that $\boldsymbol{k}$ can be in $\mathcal{K}$ or not. Now, we must transform this test into an algorithm for recovering $\mathcal{K}$.

4.3 Recovering $\mathcal{K}$

We are looking for $\dim(\mathcal{K})$ independent vectors that generate $\mathcal{K}$. Let us define $\alpha = \Pr[T(\boldsymbol{k}) = 0]$ and $\beta = \Pr[\boldsymbol{k} \in \mathcal{K}] = q^{-r}$. The following table summarises the distribution of the values of T applied to a random $\boldsymbol{k}$.

	$\boldsymbol{k} \in \mathcal{K}$	$\boldsymbol{k} \notin \mathcal{K}$	
$T(\boldsymbol{k}) = 0$	β	$\alpha - \beta$	α
$T(\boldsymbol{k}) = 1$	0	$1 - \alpha$	$1 - \alpha$
	β	$1 - \beta$	

In the case where $\gcd(\ell, n) = 8$ we have $\alpha - \beta \ll \beta$ and therefore the test T has almost no false positives. In the case where $\gcd(\ell, n) = 1$ we have $\beta \ll \alpha$ and therefore the test T cannot give direct proof of membership of $\mathcal{K}$. A specific algorithm to recover $\mathcal{K}$ is needed.

The property we use is the linearity of $\mathcal{K}$: if $\boldsymbol{k}, \boldsymbol{k'} \in \mathcal{K}$, then $\boldsymbol{k} + \boldsymbol{k'} \in \mathcal{K}$. Two algorithms are described below. The first algorithm uses a statistical bias for $T(\boldsymbol{k} + \boldsymbol{k'})$. The second algorithm searches some large clique in a graph. A concrete attack of the PMI cryptosystem will use a mix of both techniques.

Technique 1. The key idea is: if for many different $\boldsymbol{k'} \in \mathcal{K}$, $\boldsymbol{k} + \boldsymbol{k'}$ is in $\mathcal{K}$, then $\boldsymbol{k}$ is always in $\mathcal{K}$. Therefore, if for many different $\boldsymbol{k'}$ such that $T(\boldsymbol{k'}) = 0$, $T(\boldsymbol{k} + \boldsymbol{k'}) = 0$, then $\boldsymbol{k}$ is in $\mathcal{K}$ with high probability.

We make the hypothesis that for any fixed value $\boldsymbol{k}$ and random value $\boldsymbol{k'}$ the probability that $T(\boldsymbol{k} + \boldsymbol{k'}) = 0$ is independent of the probability that $T(\boldsymbol{k'}) = 0$. Under this hypothesis, we compute $p(\boldsymbol{k}) = \Pr[T(\boldsymbol{k} + \boldsymbol{k'}) = 0 \,/\, T(\boldsymbol{k'}) = 0]$.

For a random $\boldsymbol{k}$, the value $\boldsymbol{k} + \boldsymbol{k'}$ when $T(\boldsymbol{k'}) = 0$ is uniformly distributed and $p(\boldsymbol{k}) = \alpha$. However, if $\boldsymbol{k} \in \mathcal{K}$, then one can write $p(\boldsymbol{k}) = \Pr[\boldsymbol{k'} \in \mathcal{K} \,/\, T(\boldsymbol{k'}) = 0] + \Pr[\boldsymbol{k'} \notin \mathcal{K} \,/\, T(\boldsymbol{k'}) = 0].\Pr[T(\boldsymbol{k} + \boldsymbol{k'}) = 0 \,/\, \boldsymbol{k} + \boldsymbol{k'} \notin \mathcal{K}] = \frac{\beta}{\alpha} + \frac{\alpha - \beta}{\alpha}\frac{\alpha - \beta}{1 - \beta}$.

Under the hypothesis that $\beta \ll \alpha$, if $\boldsymbol{k} \in \mathcal{K}$ then $p(\boldsymbol{k})/\alpha = \frac{(1-\beta/\alpha)^2}{1-\beta} + \frac{\beta}{\alpha^2} \simeq 1 + \beta(\alpha^{-1} - 1)^2$. Therefore the difference between the values of $p(\boldsymbol{k})$ depending on whether $\boldsymbol{k} \in \mathcal{K}$ or not is of the order of $\alpha\beta$ and, by taking $N = 1/(\alpha\beta)^2$ elements $\boldsymbol{k'}$ such that $T(\boldsymbol{k'}) = 0$ and computing the average of $T(\boldsymbol{k} + \boldsymbol{k'})$, we can decide whether $\boldsymbol{k} \in \mathcal{K}$ or not. The complexity of this test is about β^{-2}.

We checked experimentally this hypothesis, for the parameters $\ell = 41$, $n = 137$ and $r = 6$. Testing if $p(\boldsymbol{k})/\alpha - 1 > \frac{1}{2}\beta(\alpha^{-1} - 1)^2$ is not sufficient to have an error-free test of membership of $\mathcal{K}$. However, testing if $p(\boldsymbol{k})/\alpha - 1 > \beta(\alpha^{-1} - 1)^2$ appear to be sufficient to detect about half of the members of $\mathcal{K}$.

Each value $\boldsymbol{k}$ has a probability q^{-r} of being in $\mathcal{K}$ and we need n distinct elements of $\mathcal{K}$. The whole complexity for finding $\mathcal{K}$ is nq^{3r}.

Technique 2. In this technique, we define a graph whose vertices are the elements $\boldsymbol{k}$ such that $T(\boldsymbol{k}) = 0$, *i.e.* elements that may be in the kernel. For each

pair $(\boldsymbol{k}, \boldsymbol{k'})$ of vertices, we compute $T(\boldsymbol{k} + \boldsymbol{k'})$. If the result is 0, then we put an edge between these two vertices. All vertices such that $\boldsymbol{k} \in \mathcal{K}$ are connected, i.e. the elements of $\mathcal{K}$ are in a large clique.

In practice, we don't construct the whole graph. We construct its restriction to N vertices. We are looking for vertices that correspond to $n - r$ independent elements of $\mathcal{K}$. If $N > n/\beta$, it is likely that the graph contains such vertices. The clique containing the elements of $\mathcal{K}$ contains at least βN vertices. Under the same hypothesis as above, that the probability that $T(\boldsymbol{k} + \boldsymbol{k'}) = 0$ is independent of the probability that $T(\boldsymbol{k}) = 0$, this graph restricted to N vertices has αN^2 edges. Apart from the vertices that correspond to elements of $\mathcal{K}$, the edges are randomly distributed. General results on random graph [2] gives us that the expected number of vertex in the clique of maximal order in random graph of N vertex with a probability α between each edge is $\frac{2 \log N}{\log 1/\alpha} + O(\log \log N)$. Therefore, if βN is significantly greater than $\frac{2 \log N}{\log 1/\alpha}$, then there will be a unique large clique, that gives a basis of $\mathcal{K}$. When $\beta \ll \alpha$, this condition is equivalent to $N \approx \beta^{-1} \log \beta^{-1}$ and the whole complexity for finding $\mathcal{K}$ is $q^{2r} \log^2 q^r$.

However, although this technique seems to be better than the previous one, we do not know a max-clique algorithm that benefits from the fact that we have a random and dense graph which has a very large clique. In practice, as we said before, a concrete attack of the PMI cryptosystem will use a mix of technique 1 (to find some elements very likely to be members of $\mathcal{K}$) and technique 2 (to extract from them a large clique).

4.4 Recovering the Plaintext

Assume we have correctly found the kernel $\mathcal{K}$. Now, we have to reconstruct a family of n bilinear equations in the $\boldsymbol{x}$ and $\boldsymbol{y}$ variable for each affine subspace parallel to $\mathcal{K}$. When this has been done, then for fixed $\boldsymbol{y}$ we can try to solve each system in the $\boldsymbol{x}$ unknowns and decide the correct solution using redundancy.

The question one may ask is whether we still find $n - \gcd(\ell, n)$ independent equations for each affine subspace. What can be said is that the original n equations from the MI scheme are clearly friend when $\boldsymbol{x}$ is restricted to a subspace. Accordingly the number of independent equations can only increase, which is in favour of the attacker. Now, given a ciphertext $\boldsymbol{y}$, its corresponding plaintext is in some subspace parallel to $\mathcal{K}$ and for such ciphertext, each family of equations allow to recover at least $n - \gcd(\ell, n)$ coordinates of $\boldsymbol{x}$. Finally, an exhaustive search allows us to find the missing coordinates in time $q^{\gcd(\ell,n)}$ as well as the correct subspace to choose.

5 Alternative Attack Against the MI Scheme

In this section, we show a new attack against the MI scheme. We apply the same technique as in the PMI scheme. First of all, we compute the differential and next we study the kernel of the transpose of this application. In order to simplify

the exposition of the attack, we assume in the following that $\gcd(\ell, n) = 1$ and $q = 2$.

5.1 Overview

As for Patarin's attack, this attack tries to find n bilinear forms in the ciphertext coordinates and in a vector related to the plaintext. Next, when a ciphertext is given, the n linear equations in the vector related to the plaintext allow us to recover this vector. Finally, since this vector is related to the plaintext by a linear system, we can easily decrypt.

More precisely, the attacks computes two bilinear systems, $C(\boldsymbol{x}, \boldsymbol{y})$ and $D(\boldsymbol{x}, \boldsymbol{y})$, such that for ${\boldsymbol{f_k}}^\top$ in the kernel of ${\boldsymbol{L_{E,k}}}^\top$ we have

$$C(\boldsymbol{E}(\boldsymbol{k}), \boldsymbol{f_k}) = 0 \text{ and } D(\boldsymbol{k}, \boldsymbol{f_k}) = 0$$

This allows to compute $\boldsymbol{k}$ from $\boldsymbol{E}(\boldsymbol{k})$.

5.2 Description

For the MI scheme, the differential can be written as

$$L_{F,k}(x) = x^{q^\ell} \cdot k + x \cdot k^{q^\ell} = k^{q^\ell+1} \cdot \left(\frac{x}{k} + \left(\frac{x}{k} \right)^{q^\ell} \right)$$

If we define the following three *linear* functions over $\mathbb{F}_{q^n}$: $\mu_k(x) = F(k) \cdot x$, where $F(k) = k^{q^\ell+1}$, $\psi(x) = x^{q^\ell} + x$ and $\theta_k(x) = \frac{x}{k}$, then

$$L_{F,k} = \mu_k \circ \psi \circ \theta_k$$

Let us define $\boldsymbol{\mu_k} = \pi \circ \mu_k \circ \pi^{-1}$, $\boldsymbol{\psi} = \pi \circ \psi \circ \pi^{-1}$ and $\boldsymbol{\theta_k} = \pi \circ \theta_k \circ \pi^{-1}$ for $k = \pi^{-1}(\boldsymbol{S}(\boldsymbol{k}))$. Therefore

$$\boldsymbol{L_{E,k}} = \boldsymbol{T} \circ \boldsymbol{\mu_k} \circ \boldsymbol{\psi} \circ \boldsymbol{\theta_k} \circ \boldsymbol{S}$$

where all terms are linear functions of $\mathbb{F}_q^n$. The matrix of $\boldsymbol{L_{F,k}}$ is a product of $n \times n$ matrices of $\mathbb{F}_q$.

Let ${\boldsymbol{f_k}}^\top$ be in the kernel of the transpose ${\boldsymbol{L_{E,k}}}^\top$. This means that the product $(\boldsymbol{f_k})(\boldsymbol{L_{E,k}})$ is the null vector $\mathbf{0}$, which is equivalent to

$$(\boldsymbol{f_k})(\boldsymbol{T}.\boldsymbol{\mu_k}.\boldsymbol{\psi}.\boldsymbol{\theta_k}.\boldsymbol{S}) = \mathbf{0}$$

where $\boldsymbol{f_k}$ is a n-dimensional row vector and $\boldsymbol{T}$, $\boldsymbol{\mu_k}$, $\boldsymbol{\theta_k}$, and $\boldsymbol{S}$ are $n \times n$ invertible matrices and $\boldsymbol{\psi}$ is a $n \times n$ matrix. Since $\boldsymbol{\theta_k}$ and $\boldsymbol{S}$ are one-to-one, this is equivalent to $(\boldsymbol{f_k})(\boldsymbol{T}.\boldsymbol{\mu_k}) \in \operatorname{Ker} \boldsymbol{\psi}^\top$, the application $\boldsymbol{\psi}^\top$ being independent of $\boldsymbol{k}$.

Recall that in the case where $\gcd(\ell, n) = 1$ the kernel of $\boldsymbol{L_{E,k}}$ is of dimension 1 and is generated by $\boldsymbol{k}$. The transpose ${\boldsymbol{L_{E,k}}}^\top$ also has a kernel of dimension 1. The kernel of $\boldsymbol{\psi}^\top$ is one-dimensional and independent of $\boldsymbol{k}$. Therefore if $q = 2$,

$\mathrm{Ker}\, \boldsymbol{\psi}^\top = \{\mathbf{0}, \hat{\boldsymbol{f}}\}$ and the previous equation can be rewritten as $(\boldsymbol{f_k})(\boldsymbol{T}.\boldsymbol{\mu_k}) = (\hat{\boldsymbol{f}})$.

From $\mu_k(x) = F(k) \cdot x$, we deduce that $\boldsymbol{\mu_k}$ is linear in

$$F(k) = F(\pi^{-1}(\boldsymbol{S}(\boldsymbol{k}))) = \pi^{-1}(\boldsymbol{T}^{-1}(\boldsymbol{E}(\boldsymbol{k})))$$

i.e. linear in $\boldsymbol{E}(\boldsymbol{k})$, and therefore the equation $(\boldsymbol{f_k})(\boldsymbol{T}.\boldsymbol{\mu_k}) = (\hat{\boldsymbol{f}})$ is bilinear in $\boldsymbol{f_k}$ and $\boldsymbol{E}(\boldsymbol{k})$. Accordingly whenever a ciphertext $\boldsymbol{y} = \boldsymbol{E}(\boldsymbol{k})$ is given, the corresponding $\boldsymbol{f_k}$ can be found by solving a linear system.

Finally, as $(\boldsymbol{f_k})(\boldsymbol{L_{E,k}}) = 0$ and $\boldsymbol{L_{E,k}}$ is linear in the $\boldsymbol{k}$ variable we have again a bilinear relation between $\boldsymbol{k}$ and $\boldsymbol{f_k}$. Now, since $\boldsymbol{f_k}$ is known from $\boldsymbol{y}$, we get a system with n equations in n coordinates of the variable $\boldsymbol{k}$. This system has a kernel of dimension one, and consequently, we can easily decrypt.

Acknowledgement

This work is supported in part by the French government through X-Crypt and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

References

1. E. Bach and J. Shallit. *Algorithmic Number Theory.* MIT Press, 1996. Volume 1 - Efficient Algorithms.
2. B. Bollobás. *Random Graphs.* Cambridge University Press, 2001. Second Edition.
3. H. Cohen. *A Course in Computational Algebraic Number Theory.* Graduate Texts in Mathematics 138. Springer-Verlag, 1993.
4. N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Eurocrypt '00*, LNCS 1807, pages 392–407. Springer-Verlag, 2000.
5. Nicolas T. Courtois. The security of Hidden Field Equations (HFE). In David Naccache, editor, *Proceedings of CT-RSA'01*, number 2020 in LNCS, pages 266–281. Springer-Verlag, 2001.
6. Nicolas T. Courtois, Magnus Daum, and Patrick Felke. On the security of HFE, HFEv- and Quartz. In Yvo Desmedt, editor, *Proceedings of Public Key Cryptography – PKC'03*, number 2567 in LNCS, pages 337–350. Springer-Verlag, 2003. Also available at `http://eprint.iacr.org/2002/138/`.
7. J. Ding. A New Variant of the Matsumoto-Imai Cryptosystem through Perturbation. In *PKC '04*, LNCS 2947, pages 305–318. Springer-Verlag, 2004.
8. Y. Ding-Feng, L. Kwok-Yan, and D. Zong-Duo. Cryptanalysis of "2R" Schemes. In *Crypto '99*, LNCS 1666, pages 315–325. Springer-Verlag, 1999.
9. J.-C. Faugère and A. Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In *Crypto '03*, LNCS 2729, pages 44–60. Springer-Verlag, 2003.
10. H. Gilbert and M. Minier. Cryptanalysis of SFLASH. In *Eurocrypt '02*, LNCS 2332, pages 288–298. Springer-Verlag, 2002.

11. A. Kipnis and A. Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In *Crypto '99*, LNCS 1666, pages 19–30. Springer-Verlag, 1999.
12. T. Matsumoto and H. Imai. Public Quadratic Polynomial-tuples for Efficient Signature-Verification and Message-Encryption. In *Eurocrypt '88*, LNCS 330, pages 419–453. Springer-Verlag, 1988.
13. J. Patarin. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt '98. In *Crypto '95*, LNCS 963, pages 248–261. Springer-Verlag, 1995.
14. J. Patarin. Assymetric Cryptography with a Hidden Monomial. In *Crypto '96*, LNCS 1109, pages 45–60. Springer-Verlag, 1996.
15. J. Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomial (IP): Two New Families of Asymmetric Algorithms. In *Eurocrypt '96*, LNCS 1070, pages 33–46. Springer-Verlag, 1996.
16. J. Patarin, L. Goubin, and N. Courtois. C^*_{-+} and HM: Variations around Two Schemes of T.Matsumoto and H.Imai. In *Asiacrypt '98*, LNCS 1514, pages 35–50. Springer-Verlag, 1998.
17. Jacques Patarin. Hidden field equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In Ueli Maurer, editor, *Proceedings of Eurocrypt'96*, number 1070 in LNCS, pages 33–48. Springer-Verlag, 1996.

A Some Useful Mathematical Results

Lemma 4. *For any integers q, i and n,* $\gcd(q^n - 1, q^i - 1) = q^{\gcd(n,i)} - 1$

Proof. Let $(r_k)_{k\geq 0}$ be the sequence of integers obtained by the Euclidean algorithm from $r_0 = n$ and $r_1 = i$. If k_0 is the largest integer such that $r_{k_0} \neq 0$, then $r_{k_0} = \gcd(n, i)$.

Similarly, let $(R_k)_{k\geq 0}$ be the sequence of polynomials obtained from the Euclidean algorithm from $R_0 = X^n - 1$ and $R_1 = X^i - 1$. We recall that n_1 is the largest integer such that $R_{n_1} = \gcd(X^n - 1, X^i - 1)$. We show by recurrence on n that for $0 \leq k \leq k_0 + 1$, $R_k = X^{r_k} - 1$. It is correct by assumption for $k = 0$ and $k = 1$. Assuming that $k \geq 2$ and $k \leq k_0 + 1$. Let us write $r_{k-2} = \alpha r_{k-1} + r_k$. Then,

$$X^{r_{k-2}} - 1 = (X^{r_{k-1}} - 1)(X^{r_{k-2}-r_{k-1}} + X^{r_{k-2}-2r_{k-1}} + \cdots + X^{r_{k-2}-\alpha r_{k-1}}) + X^{r_k} - 1$$

Therefore, $X^{r_k} - 1$ is the remainder of the division of $R_{k-2} = X^{r_{k-2}} - 1$ by $R_{k-1} = X^{r_{k-1}} - 1$ since $r_k < r_{k-1}$. So, $R_{k_0+1} = X^0 - 1 = 0$ and $R_{k_0} \neq 0$. Consequently, $k_1 = k_0$ and $R_{k_1} = R_{k_0} = X^{r_{k_0}} - 1 = X^{\gcd(i,n)} - 1$. If we replace X by q, we get the lemma.

The following lemma is useful to exactly estimate the kernel dimension. We require exact value and not upper bounds on the number of solutions as done in [13].

Lemma 5. *In a finite field $\mathbb{F}_{q^n}$ with q^n elements, the equation $X^j = A$ has either* 0 *solution or* $\gcd(j, q^n - 1)$ *solutions.*

Proof. The multiplicative group of the finite field $\mathbb{F}_{q^n}$ has $q^n - 1$ elements. The simple case is when $\gcd(j, q^n - 1) = 1$. Therefore, j is invertible modulo $(q^n - 1)$ and we denote by h the inverse of j. Then, if we raise the equation $X^j = A$ to the power h, we get $X = X^{jh} = A^h = A'$, and so there is only one solution.

On the other hand, if $\gcd(j, q^n - 1) = d \neq 1$. Let $j' = j/d$, then $\gcd(j', q^n - 1) = 1$ and let h' be the inverse of j' modulo $q^n - 1$. We can raise the equation to the power h' and get $X^d = X^{jh'} = X^{j'dh'} = A^{h'} = A'$. This equation may have no solution if A' is not a d-th power of some value of $\mathbb{F}_{q^n}$. We now show that the equation $X^d = A'$ has d solutions when A' is a d-th power. We know that there is at least one solution which can be found by a randomised algorithm of Adleman, Manders and Miller [1]. The other solutions are obtained by multiplying the original solution by the d roots of unity. We finally explain why there are d d-th roots of unity. Since the multiplicative group of a finite field is a cyclic group, there is a primitive element g, that generates the whole group. Therefore, $g' = g^{\frac{q^n-1}{d}}$ is a d-th root of unity and for $0 \leq i < d$, g'^i ranges over the set of all roots. This completes the proof of the lemma.

If $j = q^i - 1$, then we can combine both lemmas. In a finite field $\mathbb{F}_{q^n}$ with q^n elements, the equation $X^{q^i-1} = A$ has either 0 solution or $q^{\gcd(i,n)} - 1$ solutions.

A Fast Cryptanalysis of the Isomorphism of Polynomials with One Secret Problem

Ludovic Perret

ENSTA, UMA, 32 Boulevard Victor,
75739 Paris Cedex 15, France
lperret@ensta.fr

Abstract. At Eurocrypt'96, Patarin proposed [9] new cryptographic schemes based on the *Isomorphism of Polynomials with one Secret* problem (IP1S) [9]. We study in this paper a restriction of IP1S called *Polynomial Linear Equivalence* problem (PLE) [7]. We show that PLE is in fact not a restriction of IP1S, in the sense that any algorithm solving PLE can be efficiently transformed into an algorithm for solving IP1S. Motivated by the cryptanalysis of schemes based on IP1S, we present a new efficient algorithm for solving PLE. This algorithm is mainly based on a differential property of PLE. The main advantage of this approach is to translate PLE into a simple linear algebra problem. The performances of our algorithm evidence that, with the parameters proposed in [9], schemes based on IP1S are far from achieving the security level required for cryptographic applications.

Keywords: Cryptanalysis, Isomorphism of Polynomials with One Secret (IP1S), Polynomial Linear Equivalence (PLE), Jacobian Matrix.

1 Introduction

IP1S has been originally introduced by Patarin [9] to circumvent the problem of practicality encountered when using the *Graph Isomorphism* problem as an underlying problem for zero-knowledge authentication protocols [4].

IP1S can be outlined as follows: given multivariate polynomials $\big(a_1(x_1 \ldots, x_n), \ldots, a_u(x_1 \ldots, x_n)\big)$ and $\big(b_1(x_1 \ldots, x_n), \ldots, b_u(x_1 \ldots, x_n)\big)$ over $\mathbb{F}_q[x_1, \ldots, x_n]$, find - if any - an invertible matrix $S \in GL_n(\mathbb{F}_q)$ and a vector $\underline{T} \in \mathbb{F}_q^n$, such that:

$$b_i(x_1 \ldots, x_n) = a_i\big((x_1 \ldots, x_n)S + \underline{T}\big), \text{for all } i, 1 \leq i \leq u.$$

In other words, Graphs have been replaced by multivariate polynomials and permutations by bijective affine mappings. A new authentication protocol, based on IP1S, as well as a public key signature scheme were then designed in [9]. The main motivation of this paper is to study, from both a theoretical and practical point of view, the security of these schemes. To do so, we address here a relevant variant of it. The problem we call *Polynomial Linear Equivalence* problem (PLE) [7], which is the restriction of IP1S to bijective linear mappings. We stress that

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 354–370, 2005.

this is in fact not a restriction since we prove in this paper that IP1S and PLE are equivalent, in the sense that any algorithm solving PLE can be efficiently transformed into an algorithm for solving IP1S.

1.1 Previous Work

To the best of our knowledge, the first algorithm presented for IP1S is due to Geiselmann, Meier and Steinwandt [3]. We here briefly recall its principle and refer the reader to the original paper for a detailed description.
Let $\big((a_1,\ldots,a_u),(b_1,\ldots,b_u)\big) \in \mathbb{F}_q[x_1\ldots,x_n]^u \times \mathbb{F}_q[x_1\ldots,x_n]^u$, and $(S,\underline{T}) \in GL_n(\mathbb{F}_q) \times \mathbb{F}_q^n$ such that:

$$b_i(x_1\ldots,x_n) = a_i\big((x_1\ldots,x_n)S + \underline{T}\big), \text{ for all } i, 1 \leq i \leq u.$$

Moreover, let $\underline{e_j} \in \mathbb{F}_q^n$ be the vector with its jth component equal to one and zero otherwise. The main idea is to remark that if $\underline{\ell_j} \in \mathbb{F}_q^n$ is the jth row of the matrix S, then:

$$b_i(\underline{e_j}) = a_i\big(\underline{\ell_j} + \underline{T}\big), \text{ for all } i, 1 \leq i \leq u.$$

When $\underline{T} \in \mathbb{F}_q^n$ is given, an exhaustive search among $\mathbb{F}_q^n$ is then performed to recover:

$$L_j = \{\underline{\ell} \in \mathbb{F}_q^n : b_i(\underline{e_j}) = a_i(\underline{\ell} + \underline{T}), \text{ for all } i, 1 \leq i \leq u\},$$

which is a set of candidate vectors for the jth row of S.

Soon after, Levy-dit-Vehel and Perret in [7] have remarked that the jth row of S is a zero of the following system of non-linear equations:

$$\big\{a_1(\underline{x} + \underline{T}) - b_1(\underline{e_j}) = 0, \cdots, a_u(\underline{x} + \underline{T}) - b_u(\underline{e_j}) = 0\big\}. \tag{1}$$

Therefore, the set L_j of candidates for the jth row of S is equal to the set of zeroes of (1). Hence, they have substituted the exhaustive search of the elements of L_j by the computation of a Gröbner basis [7]. In this work, we use very basic tools of linear algebra for solving IP1S.

1.2 Organization of the Paper and Main Results

The paper is organized as follows. We begin in Section 2 by introducing our notations and defining more formally the PLE and IP1S problems, which are the main concern of this paper.

In Section 3, we prove that PLE is equivalent to IP1S, i.e. any algorithm solving PLE can be efficiently transformed into an algorithm for solving IP1S.

In Section 4, differential properties of PLE are presented. These properties give a strong relation between the Jacobian matrices of an instance of PLE and solutions of this problem. We also show that structural properties of PLE can be used to obtain linear equations in the components of a solution of PLE.

A new algorithm for solving PLE is described in Section 5. Using properties of section 4, we show that a partial knowledge of a solution allows us to recover it entirely by solving a suitable linear system of equations. It appears that the algorithm presented in this section is much more efficient than algorithms previously proposed [3, 7]. This is illustrated in the last part of this paper by giving experimental results obtained with our algorithm.

2 Preliminaries

2.1 Notations

We introduce in this part the notations used throughout this paper. We denote by $\mathbb{F}_q$, the finite field with $q = p^r$ elements (p a prime, and $r \geq 1$), by $\underline{x}$ the vector $(x_1, \ldots, x_n)$, by $\mathbb{F}_q[\underline{x}] = \mathbb{F}_q[x_1, \ldots, x_n]$, the polynomial ring in the n indeterminate $x_1, \ldots, x_n$ over $\mathbb{F}_q$, and $f(\underline{x})$ stands for $f(x_1, \ldots, x_n)$. Moreover, let g and $h_1, \ldots, h_n$ be polynomials of $\mathbb{F}_q[\underline{x}]$; by $g \circ \underline{h}$ we shall mean the functional composition $g\big(h_1, \cdots, h_n\big)$ of g and the h_i's.

A *monomial* is a power product of the variables $x_1, \ldots, x_n$, and a *term* is a coefficient multiplied by a monomial. We shall define the *total degree* of a monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n}, (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, by the sum $\sum_{i=1}^n \alpha_i$. Obviously, the *total degree* of a term $cx_1^{\alpha_1} \cdots x_n^{\alpha_n}, c \in \mathbb{F}_q^*$, is the total degree of $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. The *leading term* of f is the largest term among the terms of f w.r.t. some admissible ordering on the monomials. For example, the lexicographical order $\prec_{LEX}$, defined by:

$$x_1^{\alpha_1} \cdots x_n^{\alpha_n} \prec_{LEX} x_1^{\beta_1} \cdots x_n^{\beta_n} \Longleftrightarrow \begin{cases} \text{the first coordinates } \alpha_i \text{ and } \beta_i \text{ from the left} \\ \text{which are different satisfy } \alpha_i < \beta_i, \end{cases}$$

is an admissible order.

Let $f \in \mathbb{F}_q[\underline{x}]$, the *degree* of f is the total degree of its leading term. We shall say that f is *homogeneous* of degree d if every term appearing in f has total degree d. An important fact is that every polynomial can be written uniquely as a sum of homogeneous polynomials. Namely $f = \sum_d f^{(d)}$, with $f^{(d)}$ being the sum of all terms of f of total degree d. Notice that each $f^{(d)}$ is homogeneous, and we call $f^{(d)}$ the dth *homogeneous component of* f. If f is of maximal total degree d, we shall call *homogenization* of f, denoted by F, the polynomial:

$$F(x_1, \ldots, x_n, z) = \sum_{i=0}^{d} f^{(i)}(x_1, \ldots, x_n) z^{d-i}. \tag{2}$$

The polynomials f and F are related in the following way:

$$F(\underline{x}, z) = z^d f\big(\frac{x_1}{z}, \ldots, \frac{x_n}{z}\big) = z^d f\big(\frac{\underline{x}}{z}\big). \tag{3}$$

Evaluating F in $(\underline{x}, 1)$ yields f, i.e. $F(\underline{x}, 1) = f(\underline{x})$. This process is called *dehomogenization*.

We extend now some of the notations previously given to vectors of polynomials. Precisely, for $\underline{a} = \big(a_1, \cdots, a_u\big) \in \mathbb{F}_q[\underline{x}]^u$, we shall denote by $\underline{a}^{(d)} = (a_1^{(d)}, \ldots, a_u^{(d)})$ the dth homogeneous components of the polynomials of $\underline{a}$.

We shall denote by $\mathcal{M}_{n,u}(\mathbb{F}_q)$ the set of $n \times u$ matrices whose components are in $\mathbb{F}_q$. For $M \in \mathcal{M}_{n,u}(\mathbb{F}_q)$, we set $Ker(M) = \{\underline{x} \in \mathbb{F}_q^n : \underline{x}M = \underline{0_u}\}$, $\underline{0_u}$ being the null vector of $\mathbb{F}_q^u$. As usual, $GL_n(\mathbb{F}_q)$ denotes the set of invertible matrices of $\mathcal{M}_{n,n}(\mathbb{F}_q)$, and we denote by $AGL_n(\mathbb{F}_q)$ the cartesian product $GL_n(\mathbb{F}_q) \times \mathbb{F}_q^n$.

2.2 Jacobian Matrix

Let $f = \sum_i a_i x^i \in \mathbb{F}_q[x]$, the *formal derivative* of f is the polynomial $\frac{df}{dx} = \sum_i i a_i x^{i-1} \in \mathbb{F}_q[x]$. More generally, when $f \in \mathbb{F}_q[x_1, \ldots, x_n]$, the *partial derivatives* of f, denoted by $\frac{\partial f}{\partial x_i}$, $1 \leq i \leq n$, are defined by considering f as a polynomial in x_i with coefficients in $\mathbb{F}_q[x_1 \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$. It is not hard to check that the $\partial/\partial x_i$'s commute with one another.

Definition 1. *The* Jacobian matrix *of* $\underline{f} = (f_1, \ldots, f_u) \in \mathbb{F}_q[\underline{x}]^u$*, denoted by* $J_{\underline{f}}(\underline{x})$*, is the* $u \times n$ *matrix whose components are the partial derivatives of the polynomials of* $\underline{f}$*, i.e.:*

$$J_{\underline{f}}(\underline{x}) = \left\{ \frac{\partial f_i}{\partial x_j}(\underline{x}) \right\}_{1 \leq j \leq n}^{1 \leq i \leq u}$$

The property of partial derivatives that we use in this paper is the chain rule condition:

$$\frac{\partial(g \circ \underline{h})}{\partial x_i}(\underline{x}) = \sum_{j=1}^{n} \frac{\partial g}{\partial x_j}(\underline{h}(\underline{x})) \frac{\partial h_j}{\partial x_i}(\underline{x}), \text{for all } i, 1 \leq i \leq n.$$

2.3 The IP1S and PLE Problems

Let $\big(\underline{a} = (a_1, \ldots, a_u), \underline{b} = (b_1, \ldots, b_u)\big) \in \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u$. We shall say that $(\underline{a}, \underline{b})$ are *affine-equivalent*, denoted by $\underline{a} \equiv_A \underline{b}$, if there exists $(S, \underline{T}) \in AGL_n(\mathbb{F}_q)$, s.t.:

$$b_i(x_1 \ldots, x_n) = a_i\big((x_1 \ldots, x_n)S + \underline{T}\big), \text{for all } i, 1 \leq i \leq u.$$

We call such a pair an *affine equivalence pair*. The *Isomorphism of Polynomials with one Secret* problem (IP1S) is then the one of finding - if any - an affine equivalence pair between the polynomials of $\underline{a}$ and $\underline{b}$. We mention that this problem is also called *Polynomial Affine Equivalence* problem (PAE) in [7].

A natural variant of this problem is to consider linear bijective mappings.

We shall say that $(\underline{a}, \underline{b})$ are *linear-equivalent*, denoted by $\underline{a} \equiv_L \underline{b}$, if there exists $S \in GL_n(\mathbb{F}_q)$, such that:

$$b_i(\underline{x}) = a_i(\underline{x}S), \text{for all } i, 1 \leq i \leq u. \tag{4}$$

In the sequel we shall denote, for convenience, equations (4) by $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$.

We call the matrix S a *linear equivalence matrix*. The *Polynomial Linear Equivalence* problem (PLE) is then the one of finding - if any - a linear equivalence matrix between $\underline{a}$ and $\underline{b}$.

3 IP1S and PLE are Equivalent

Before giving our complexity results, we need to present structural properties of PLE and IP1S.

Property 1. *If $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S + \underline{T})$, for some $(S, \underline{T}) \in AGL_n(\mathbb{F}_q)$, then:*

$$b_i^{(D_i)}(\underline{x}) = a_i^{(D_i)}(\underline{x}S), \text{for all } i,\ 1 \leq i \leq u,$$

D_i being, for all i, $1 \leq i \leq u$, the degree of the homogeneous component of highest degree of b_i.

Proof. For all i, $1 \leq i \leq u$, $b_i(\underline{x}) = a_i(\underline{x}S + \underline{T})$, for some $(S, \underline{T}) \in AGL_n(\mathbb{F}_q)$ implies that $b_i(\underline{x} - \underline{T}S^{-1}) = a_i(\underline{x}S)$. We stress that $b_i^{(D_i)}(\underline{x} - \underline{T}S^{-1})$, which is the homogeneous component $b_i^{(D_i)}$ of b_i evaluated in $\underline{x} - \underline{T}S^{-1}$, contains the terms of total degree D_i of $b_i(\underline{x} - \underline{T}S^{-1})$.
Indeed, let $b_i^{(D_i)}(\underline{x}) = \sum_{1 \leq j_1, \ldots, j_{D_i} \leq n} b_{i,j_1,\ldots,j_{D_i}}^{(D_i)} x_{j_1} \cdots x_{j_{D_i}}$, be the homogeneous component of degree D_i of b_i. Since:

$$\prod_{k=1}^{D_i} \left(x_{j_k} - (\underline{T}S^{-1})_{j_k}\right) = \underbrace{x_{j_1} \cdots x_{j_{D_i}}}_{\text{total degree } D_i} + \text{terms of total degree} < D_i.$$

We have:

$$\begin{aligned} b_i^{(D_i)}(\underline{x} - \underline{T}S^{-1}) &= \sum_{1 \leq j_1, \ldots, j_{D_i} \leq n} b_{i,j_1,\ldots,j_{D_i}}^{(D_i)} \prod_{k=1}^{D_i} \left(x_{j_k} - (\underline{T}S^{-1})_{j_k}\right) \\ &= \underbrace{b_i^{(D_i)}(\underline{x})}_{\text{total degree } D_i} + \text{terms of total degree} < D_i. \end{aligned}$$

Finally, by equating the terms of total degree D_i of $b_i(\underline{x} - \underline{T}S^{-1})$ with those of $a_i(\underline{x}S)$, we get that $b_i^{(D_i)}(\underline{x}) = a_i^{(D_i)}(\underline{x}S)$, for all $i, 1 \leq i \leq u$. □

Remark 1. *Let $\left(\underline{a} = (a_1, \ldots, a_u), \underline{b} = (b_1, \ldots, b_u)\right) \in \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u$. In the rest of the paper, D_i will always denote the degree of the homogeneous component of highest degree of b_i. Moreover, we set $D = max_{1 \leq i \leq u}(D_i)$.*

We now give the linear counterpart of property 1. Remark that the next property already appeared in [7], but is quoted here for the sake of completeness.

Property 2. *Let $S \in GL_n(\mathbb{F}_q)$, we have:*

$$\underline{b}(\underline{x}) = \underline{a}(\underline{x}S) \Longleftrightarrow \underline{b}^{(d)}(\underline{x}) = \underline{a}^{(d)}(\underline{x}S), \text{for all } d, 0 \leq d \leq D.$$

Proof. Let $S \in GL_n(\mathbb{F}_q)$, such that $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$. For each $i, 1 \leq i \leq u$, and for all d, $0 \leq d \leq D$, the terms of total degree d of $a_i(\underline{x}S)$ are equal to those of the homogeneous polynomial $a_i^{(d)}$ evaluated in $\underline{x}S$, i.e. the terms of $a_i^{(d)}(\underline{x}S)$. Thus, by equating the terms of total degree d of $b_i(\underline{x})$ with those of $a_i(\underline{x}S)$, we get that for all $i, 1 \leq i \leq u$:

$$b_i^{(d)}(\underline{x}) = a_i^{(d)}(\underline{x}S), \text{for all } d, 0 \leq d \leq D.$$

Let $S \in GL_n(\mathbb{F}_q)$ and suppose now that for all $i, 1 \leq i \leq u, b_i^{(d)}(\underline{x}) = a_i^{(d)}(\underline{x}S)$, for all $d, 0 \leq d \leq D$. Consequently, we get that $\sum_{d=0}^{D} b_i^{(d)}(\underline{x}) = \sum_{d=0}^{D} a_i^{(d)}(\underline{x}S)$, i.e. $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$. □

We now introduce some additional notations. We shall call dPLE (resp. dIP1S) the decisional version of PLE (resp. IP1S); that is, the problem of deciding whether $(\underline{a}, \underline{b}) \in \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u$ are linear-equivalent (resp. affine-equivalent).

Finally, we would like to recall that a polynomial-time many-one reduction (also known as Karp reduction) is defined as follows:

Definition 2. *[5] Let A and B be two decisional problems. A is* polynomial-time many-one reducible *to B, denoted by $A \leq_p^m B$, iff there exists a polynomial-time computable function f, such that for any instance x of A, we have:*

$$x \in L_A \iff f(x) \in L_B,$$

L_A and L_B being the set of YES instances of A and B.
Moreover, A and B are polynomial-time many-one equivalent*, denoted by $A \equiv_p^m B$, iff $A \leq_p^m B$ and $B \leq_p^m A$.*

For dIP1S and dPLE, we have the following (surprising) result:

Proposition 1. *dIP1S is polynomial-time many-one reducible to dPLE.*

Proof. In order to prove that dIP1S$\leq_p^m$dPLE, we define a function $f : \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u \to \mathbb{F}_q[\underline{x}, z]^{u+1} \times \mathbb{F}_q[\underline{x}, z]^{u+1}$ as follows. For all $(\underline{a}, \underline{b}) \in \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u$:

$$f\big(\underline{a}(\underline{x}), \underline{b}(\underline{x})\big) = \big(\underline{A}(\underline{x}, z), \underline{B}(\underline{x}, z)\big),$$

with $\underline{A}(\underline{x}, z) = \big(A_1(\underline{x}, z), \ldots, A_u(\underline{x}, z), z\big)$ and $\underline{B}(\underline{x}, z) = \big(B_1(\underline{x}, z), \ldots, B_u(\underline{x}, z), z\big)$. The A_i's (resp. B_i's) being the homogenizations of the a_i's (resp. b_i's). One can see at once that, according to (2), f can be computed in polynomial-time.

Now, let $(\underline{a}, \underline{b}) \in L_{dIP1S}$, i.e. $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S + \underline{T})$, for some $\big(S = \{s_{i,j}\}_{1 \leq i,j \leq n}, \underline{T} = (t_1, \ldots, t_n)\big) \in AGL_n(\mathbb{F}_q)$. From this affine equivalence pair, we define the following matrix:

$$S' = \begin{pmatrix} s_{1,1} & s_{1,2} & \ldots & s_{1,n} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{n,1} & s_{n,2} & \ldots & s_{n,n} & 0 \\ t_1 & t_2 & \ldots & t_n & 1 \end{pmatrix}.$$

We mention that since $S \in GL_n(\mathbb{F}_q)$, then $S' \in GL_{n+1}(\mathbb{F}_q)$. Indeed, it's inverse is $\begin{pmatrix} S^{-1} & \underline{0_n}^t \\ -TS^{-1} & 1 \end{pmatrix}$. Moreover, we have:

$$\begin{aligned} (\underline{x}, z)S' &= \big(\textstyle\sum_{j=1}^{n} x_j s_{j,1} + t_1 z, \ldots, \sum_{j=1}^{n} x_j s_{j,n} + t_n z, z\big) \\ &= \big(\underline{x}S + \underline{T}z, z\big) \end{aligned} \tag{5}$$

Recall that for all $i, 1 \leq i \leq u$, D_i denotes the degree of the homogeneous component of highest degree of b_i. Note that for all $z \neq 0$, $z^{D_i} b_i(\frac{\underline{x}}{z}) = z^{D_i} a_i(\frac{\underline{x}}{z}S + \underline{T})$. Thus, using (3) and (5), we get that for all $i, 1 \leq i \leq u$:

$$B_i(\underline{x},z) = z^{D_i} a_i\big(\frac{\underline{x}}{z}S + \underline{T}\big) = z^{D_i} a_i\big(\frac{\underline{x}S + \underline{T}z}{z}\big) = A_i\big(\underline{x}S + \underline{T}z, z\big) = A_i\big((\underline{x},z)S'\big).$$

To handle the case $z = 0$, we use property 1. According to it, we know that if $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S + \underline{T})$, for some $(S, \underline{T}) \in AGL_n(\mathbb{F}_q)$, then $b_i^{(D_i)}(\underline{x}) = a_i^{(D_i)}(\underline{x}S)$, for all i, $1 \leq i \leq u$. Therefore, for $z = 0$, and for all $i, 1 \leq i \leq u$:

$$A_i\big((\underline{x},0)S'\big) = A_i\big(\underline{x}S, 0\big) = a_i^{(D_i)}\big(\underline{x}S\big) = b_i^{(D_i)}\big(\underline{x}\big) = B_i\big(\underline{x},0\big).$$

Finally, we remark that $A_{u+1}\big((\underline{x},z)S'\big) = A_{u+1}\big(\underline{x}S + \underline{T}z, z\big) = z = B_{u+1}\big(\underline{x},z\big)$. Thus, we get that $f\big(\underline{a},\underline{b}\big) = \big(\underline{A},\underline{B}\big) \in L_{dPLE}$.

Now, let $f\big(\underline{a},\underline{b}\big) = \big(\underline{A},\underline{B}\big) \in L_{dPLE}$, i.e. $\underline{B}\big(\underline{x},z\big) = \underline{A}\big((\underline{x},z)S''\big)$, for some $S'' = \{s''_{i,j}\}_{1\leq i,j\leq n+1} \in GL_{n+1}(\mathbb{F}_q)$. Due to the particular shape of the polynomials of $\underline{A}$ and $\underline{B}$, we must have $z = \sum_{j=1}^{n} x_j s''_{j,n+1} + z s''_{n+1,n+1}$, i.e. $s''_{j,n+1} = 0$, for all $j, 1 \leq j \leq n$ and $s''_{n+1,n+1} = 1$. Thus, the linear equivalence matrix S'' must leave z unchanged. Therefore, if we set $h_1(S'') = \{s''_{i,j}\}_{1\leq i,j\leq n}$ and $h_2(S'') = (s''_{n+1,1}, \ldots, s''_{n+1,n})$ then for all $i, 1 \leq i \leq u$, we have:

$$B_i\big(\underline{x},z\big) = A_i\big((\underline{x},z)S''\big) = A_i\big(\underline{x}h_1(S'') + z h_2(S''), z\big) = z^{D_i} a_i\big(\frac{\underline{x}}{z}h_1(S'') + h_2(S'')\big).$$

For $z = 1$, we get in particular that:

$$\underline{B}(\underline{x},1) = \big(\underline{b}(\underline{x}),1\big) = \underline{A}((\underline{x},1)S'') = \big(\underline{a}(\underline{x}h_1(S'') + h_2(S''),1\big).$$

Hence, $\underline{b}(\underline{x}) = \underline{a}\big(\underline{x}h_1(S'') + h_2(S'')\big)$. Since $S'' \in GL_{n+1}(\mathbb{F}_q)$, $h_1(S'') \in GL_n(\mathbb{F}_q)$ and it follows that $\big(h_1(S''), h_2(S'')\big)$ is an affine equivalence pair between $\underline{a}$ and $\underline{b}$, i.e. $(\underline{a},\underline{b}) \in L_{dIP1S}$. □

Note that in this paper, we are interested in the finding of a solution of PLE (resp. IP1S) rather than deciding if such a solution exists. However, this result permits in fact to transform efficiently any algorithm dedicated to PLE to an algorithm for solving IP1S. Indeed, let the notations be as in the proof of proposition 1 and $(\underline{a},\underline{b})$ be an instance of IP1S. Any linear equivalence S'' for $f(\underline{a},\underline{b}) = \big(\underline{A},\underline{B}\big)$ can be efficiently transformed into an affine equivalence pair $\big(h_1(S''), h_2(S'')\big)$ for $(\underline{a},\underline{b})$. Thus, any solution given by a PLE algorithm, on input $\big(\underline{A},\underline{B}\big)$, can be easily transformed to a solution for IP1S, i.e. an affine equivalence pair for $(\underline{a},\underline{b})$.

On the other hand, we have the following (less surprising) result:

Proposition 2. *dPLE is polynomial-time many-one reducible to dIP1S.*

Proof. Let $(\underline{a},\underline{b}) \in \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u$. For proving that dPLE$\leq_p^m$dIP1S, we define $f : \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u \to \mathbb{F}_q[\underline{x}]^{(D+1)\cdot u} \times \mathbb{F}_q[\underline{x}]^{(D+1)\cdot u}$ in the following way. For all $(\underline{a},\underline{b}) \in \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u$, we have:

$$f(\underline{a},\underline{b}) = (\underline{A},\underline{B}),$$

with $\underline{A} = \left(\underline{a}^{(D)}, \underline{a}^{(D-1)}, \ldots, \underline{a}^{(0)}\right)$ and $\underline{B} = \left(\underline{b}^{(D)}, \underline{b}^{(D-1)}, \ldots, \underline{b}^{(0)}\right)$.
Let $(\underline{a},\underline{b}) \in L_{dPLE}$, i.e. $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in GL_n(\mathbb{F}_q)$.
According to property 2, we have $\underline{b}^{(d)}(\underline{x}) = \underline{a}^{(d)}(\underline{x}S)$, for all $d, 0 \leq d \leq D$. Thus $\underline{B}(\underline{x}) = \underline{A}(\underline{x}S)$, and $(S, \underline{0}_n)$ is an affine equivalence pair between $\underline{A}$ and $\underline{B}$, i.e. $f(\underline{a},\underline{b}) = (\underline{A},\underline{B}) \in L_{dIP1S}$.

Now let $f(\underline{a},\underline{b}) = (\underline{A},\underline{B}) \in L_{dIP1S}$, i.e. $\underline{B}(\underline{x}) = \underline{A}(\underline{x}S' + \underline{T}')$, for some $(S', \underline{T}') \in AGL_n(\mathbb{F}_q)$. By the very construction of f, $\underline{B}(\underline{x}) = \underline{A}(\underline{x}S' + \underline{T}')$ implies that $\underline{b}^{(d)}(\underline{x}) = \underline{a}^{(d)}(\underline{x}S' + \underline{T}')$, for all $d, 0 \leq d \leq D$. We then have according to property 1 that:

$$\underline{b}^{(d)}(\underline{x}) = \underline{a}^{(d)}(\underline{x}S'), \text{ for all } d, 0 \leq d \leq D.$$

By property 2, we get that $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S')$, i.e. S' is a linear equivalence matrix between $\underline{a}$ and $\underline{b}$, proving that $(\underline{a},\underline{b}) \in L_{dPLE}$. □

Let the notations be as in the proof of proposition 2 and $(\underline{a},\underline{b})$ be an instance of PLE. If $(S, \underline{T})$ is an affine equivalence pair, between $f(\underline{a},\underline{b}) = (\underline{A},\underline{B})$, then S is a linear equivalence matrix between $(\underline{A},\underline{B})$, and thus between $(\underline{a},\underline{b})$. Thus, from any solution given by an IP1S algorithm, on input $(\underline{A},\underline{B})$, one can easily construct a solution to PLE for $(\underline{a},\underline{b})$.
Finally, from propositions 1 and 2, we deduce:

Corollary 1. $dPLE \equiv_p^m dIP1S$.

This equivalence result also holds for PLE and IP1S (the search problems associated to dPLE and dIP1S). Indeed, aboves proofs construct a solution of PLE (resp. IP1S) from one of IP1S (resp. PLE). Thus, we can w.l.o.g restrict our attention to only one of these problems. Hereafter, we will focus on PLE. We have chosen more particularly this problem since it seems to have more useful algorithmic properties.

4 Properties of PLE

We present in this part new properties of PLE. In 4.1, we give a strong relation between the Jacobian matrices of an instance $(\underline{a},\underline{b})$ of PLE and solutions of this instance. In 4.2, we show that structural properties of PLE permit to obtain linear equations in the components of a linear equivalence matrix (provided such a matrix exists).

4.1 Differential Properties

In the one variable case (i.e. $n = 1$), PLE can be reformulated as follows: given polynomials $a_1(x), \ldots, a_u(x)$ and $b_1(x), \ldots, b_u(x)$ in $\mathbb{F}_q[x]$, find - if any - $s \in \mathbb{F}_q$,

such that the equality $b_i(x) = a_i(xs)$ holds for all $i, 1 \leq i \leq u$. When computing the formal derivatives of these equalities, we get that s must be such that:

$$\frac{db_i}{dx}(x) = s\frac{da_i}{dx}(xs), \text{ for all } i, 1 \leq i \leq u.$$

Thus, if $\frac{da_i}{dx}(0) \neq 0$, for some i, then $s = \frac{\frac{db_i}{dx}(0)}{\frac{da_i}{dx}(0)}$. The next theorem, which is is the main result of this section, extend this idea to multivariate polynomials.

Theorem 1. *If $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in GL_n(\mathbb{F}_q)$, then:*

$$J_{\underline{b}}(\underline{x}) = J_{\underline{a}}(\underline{x}S)S^t,$$

$J_{\underline{a}}(\underline{x}S) = \left\{\frac{\partial a_i}{\partial x_j}(\underline{x}S)\right\}_{1\leq j\leq n}^{1\leq i\leq u}$ and $J_{\underline{b}}(\underline{x}) = \left\{\frac{\partial b_i}{\partial x_j}(\underline{x})\right\}_{1\leq j\leq n}^{1\leq i\leq u}$ being the Jacobian matrices of $\underline{a}$ evaluated in $\underline{x}S$ and of $\underline{b}$ evaluated in $\underline{x}$, respectively.

From this theorem, we deduce the following corollaries:

Corollary 2. *Let $S \in GL_n(\mathbb{F}_q)$ be such that $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, and $(\underline{p'}, \underline{p}) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$ be such that $\underline{p'} = \underline{p}S$. Then:*

$$\begin{array}{l} i)\ J_{\underline{b}}(\underline{p}) = J_{\underline{a}}(\underline{p'})S^t \\ ii)\ Ker\big(J_{\underline{a}}^t(\underline{p'})\big) = Ker\big(J_{\underline{b}}^t(\underline{p})\big)S \end{array}$$

Proof. *i)* is obvious since $\underline{p'} = \underline{p}S$.
For *ii)*, let $\underline{k_a} \in Ker\big(J_{\underline{a}}^t(\underline{p'})\big)$, we have $\underline{k_a}S^{-1}J_{\underline{b}}^t(\underline{p}) = \underline{k_a}J_{\underline{a}}^t(\underline{p'}) = \underline{0_u}$, therefore $\underline{k_a}S^{-1} \in Ker\big(J_{\underline{b}}^t(\underline{p})\big)$, i.e. $\underline{k_a} \in Ker\big(J_{\underline{b}}^t(\underline{p})\big)S$.
Now, let $\underline{k'} = \underline{k_b}S \in Ker(J_{\underline{b}}^t(\underline{p}))S$, we have $\underline{0_u} = \underline{k_b}J_{\underline{b}}^t(\underline{p}) = \underline{k'}J_{\underline{a}}^t(\underline{p'})$, i.e. $\underline{k'} \in Ker\big(J_{\underline{a}}^t(\underline{p'})\big)$. Thus, $Ker\big(J_{\underline{a}}^t(\underline{p'})\big) = Ker\big(J_{\underline{b}}^t(\underline{p})\big)S$. □

Corollary 3. *If $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in GL_n(\mathbb{F}_q)$, then:*

$$J_{\underline{b}^{(d)}}(\underline{x}) = J_{\underline{a}^{(d)}}(\underline{x}S)S^t, \text{for all } d, 0 \leq d \leq D.$$

$J_{\underline{a}^{(d)}}(\underline{x}S)$ and $J_{\underline{b}^{(d)}}(\underline{x})$ being the Jacobian matrices of $\underline{a}^{(d)}$ evaluated in $\underline{x}S$ and of $\underline{b}^{(d)}$ evaluated in $\underline{x}$, respectively.

4.2 Structural Properties

For each homogeneous polynomial $p \in \mathbb{F}_q[\underline{x}]$ of degree two there exists $Q \in \mathcal{M}_{n,n}(\mathbb{F}_q)$, such that $p(\underline{x}) = \underline{x}Q\underline{x}^t$. This matrix can be easily constructed from the knowledge of the coefficients of the terms of p, but is not unique in general. For fields of characteristic $\neq 2$, provided that Q is symmetric (resp. upper triangular, lower triangular) such a representation is unique. For fields of characteristic 2, the representation is unique if Q is upper triangular or lower triangular.

Corollary 4. *Let $Q_{a_i}, Q_{b_i} \in \mathcal{M}_{n,n}(\mathbb{F}_q)$ be, for all $i, 1 \leq i \leq u$, the unique matrices*[1] *such that $a_i^{(2)}(\underline{x}) = \underline{x}Q_{a_i}\underline{x}^t$ and $b_i^{(2)}(\underline{x}) = \underline{x}Q_{b_i}\underline{x}^t$. If $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in GL_n(\mathbb{F}_q)$, then:*

$$i)\, Q_{b_i} = SQ_{a_i}S^t, \text{ for all } i, 1 \leq i \leq u$$
$$ii)\, Ker(Q_{a_i}) = Ker(Q_{b_i})S, \text{ for all } i, 1 \leq i \leq u.$$

Proof. For i), we obtain by property 2 that if $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in GL_n(\mathbb{F}_q)$, then $\underline{b}^{(2)}(\underline{x}) = \underline{a}^{(2)}(\underline{x}S)$. Thus, for all $i, 1 \leq i \leq u$, we have $\underline{x}Q_{b_i}\underline{x}^t = \underline{x}SQ_{a_i}S^t\underline{x}^t$, i.e. $Q_{b_i} = SQ_{a_i}S^t$.
For ii), let $\underline{k_{a_i}} \in Ker(Q_{a_i})$, we have $\underline{k_{a_i}}S^{-1}Q_{b_i} = \underline{k_{a_i}}Q_{a_i}S^t = \underline{0_n}S^t = \underline{0_n}$, thus $\underline{k_{a_i}}S^{-1} \in Ker(Q_{b_i})$, i.e. $\underline{k_{a_i}} \in Ker(Q_{b_i})S$, for all $i, 1 \leq i \leq u$.
Now, let $\underline{k'} = \underline{k_{b_i}}S \in Ker(Q_{b_i})S$, we have $\underline{0_n} = \underline{k_{b_i}}Q_{b_i}(S^t)^{-1} = \underline{k'}Q_{a_i}$, and thus $\underline{k'} \in Ker(Q_{a_i})$, for all $i, 1 \leq i \leq u$. □

We finish this part by extending, thanks to property 2, a result given in [2].

Corollary 5. *Let $Q_{a_i}, Q_{b_i} \in \mathcal{M}_{n,n}(\mathbb{F}_q)$ be, for all $i, 1 \leq i \leq u$, the unique matrices such that $a_i^{(2)}(\underline{x}) = \underline{x}Q_{a_i}\underline{x}^t$ and $b_i^{(2)}(\underline{x}) = \underline{x}Q_{b_i}\underline{x}^t$. Moreover, let $S \in GL_n(\mathbb{F}_q)$ be such that $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$. If there exists $j, 1 \leq j \leq n$, such that Q_{b_j} is invertible then for all $i, 1 \leq i \neq j \leq n$:*

$$S^tQ_{b_j}^{-1}Q_{b_i} = Q_{a_j}^{-1}Q_{a_i}S^t. \tag{6}$$

Proof. According to corollary 4, we have $Q_{b_i} = SQ_{a_i}S^t$, for all $i, 1 \leq i \leq u$. Moreover, since Q_{b_j} and S are invertible, we get that $S^{-1} = Q_{a_j}S^tQ_{b_j}^{-1}$. It follows that, for all $i, 1 \leq i \neq j \leq n$, $Q_{a_j}S^tQ_{b_j}^{-1}Q_{b_i} = Q_{a_i}S^t$. Finally, since Q_{b_j} is invertible then Q_{a_j} is also invertible and we get that $S^tQ_{b_j}^{-1}Q_{b_i} = Q_{a_j}^{-1}Q_{a_i}S^t$, for all $i, 1 \leq i \neq j \leq n$. □

We stress that this corollary extends the result given in [2], since equation (6) holds for all instances of PLE whereas the result quoted in [2] holds for instances of PLE composed of homogeneous polynomials of degree 2 only.

5 The PLE Algorithm

Levy-dit-Vehel and Perret have linked PLE with the problem of finding common zeroes of multivariate polynomials [7]. We go one step further in this section. Indeed, we show that a partial knowledge of a linear equivalence matrix allows us to recover it entirely by solving a suitable linear system of equations.

[1] In upper triangular form, lower triangular form, or symmetric form, if such a matrix exists.

5.1 Description of the PLE Algorithm

In the sequel, we always suppose that $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in GL_n(\mathbb{F}_q)$.

Let us present now the main ideas of our algorithm.

How to easily recover linear equations in the components of S?

We describe here how to obtain, from properties described in section 4, linear equations in the components of S. Indeed, let Q_{a_i} and Q_{b_i} be, for all $i, 1 \leq i \leq u$, defined as in corollary 4. By corollary 5, we have that, whenever Q_{b_j} is invertible for some $j, 1 \leq j \leq n$, then $S^t Q_{b_j}^{-1} Q_{b_i} = Q_{a_j}^{-1} Q_{a_i} S^t$, for all $i, 1 \leq i \neq j \leq n$. Moreover, according to corollary 2, we have additionally that $J_{\underline{b}}(\underline{0_n}) = J_{\underline{a}}(\underline{0_n})S^t$. Thus, S is a particular solution of the following linear system of equations, with unknowns the components of $X \in \mathcal{M}_{n,n}(\mathbb{F}_q)$:

$$\begin{cases} J_{\underline{b}}(\underline{0_n}) = J_{\underline{a}}(\underline{0_n})X^t \\ X^t Q_{b_j}^{-1} Q_{b_i} = Q_{a_j}^{-1} Q_{a_i} X^t, \forall i, j 1 \leq i \neq j \leq n, \text{ s.t. } Q_{b_j} \text{ is invertible} \end{cases} \tag{7}$$

How to start the algorithm?

In our algorithm, we need to find pairs $(\underline{p'}, \underline{p}) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$, such that $\underline{p'} = \underline{p}S$. Such a pair can obviously be recovered by randomly selecting $\underline{p} \in \mathbb{F}_q^n$ and then performing an exhaustive search, over $\mathbb{F}_q^n$, to find the corresponding vector $\underline{p'} = \underline{p}S$. In many cases, we can, thanks to properties of section 4, significantly decrease the cost of this exhaustive search.

Indeed, according to corollary 2, $Ker\big(J_{\underline{a}}^t(\underline{0_n})\big) = Ker\big(J_{\underline{b}}^t(\underline{0_n})\big)S$. Consequently, any vector $\underline{p} \in Ker\big(J_{\underline{b}}^t(\underline{0_n})\big)$ is mapped to $Ker\big(J_{\underline{a}}^t(\underline{0_n})\big)$, i.e. there exists $\underline{p'} \in Ker\big(J_{\underline{a}}^t(\underline{0_n})\big)$ such that $\underline{p'} = \underline{p}S$. Thus, if we chose a vector $\underline{p} \in Ker\big(J_{\underline{b}}^t(\underline{0_n})\big)$ then $\underline{p'} = \underline{p}S$ can be recovered by listing all elements of $Ker\big(J_{\underline{a}}^t(\underline{0_n})\big)$, rather than all $\mathbb{F}_q^n$.

Similarly, using the quadratic parts of the polynomials of $\underline{a}$ and $\underline{b}$, we obtain, according to corollary 4, that for all $i, 1 \leq i \leq u$, any vector $\underline{p} \in Ker(Q_{b_i})$ is mapped to an element of $Ker(Q_{a_i})$. Thus, by choosing $\underline{p} \in Ker(Q_{b_i})$, we can recover $\underline{p'} = \underline{p}S$ by performing an exhaustive search over $Ker(Q_{a_i})$.

How to use Jacobian matrices?

Let $(\underline{p'}, \underline{p}) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$ be such that $\underline{p'} = \underline{p}S$. According to corollary 2, we have $J_{\underline{b}}(\underline{p}) = J_{\underline{a}}(\underline{p'})S^t$. From this equality, we obtain $n \cdot u$ linear equations in n^2 unknowns (the components of S), $n \cdot Rank\big(J_{\underline{a}}(\underline{p'})\big)$ of which are linearly independent. When $Rank\big(J_{\underline{a}}(\underline{p'})\big) < n$, all the solutions found do not necessarily give a linear equivalence matrix between $\underline{a}$ and $\underline{b}$. To eliminate superfluous solutions, we need to find new linear equations in the components of S. To do so, we increase the number of pairs $(\underline{p'}, \underline{p}) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$, such that $\underline{p'} = \underline{p}S$. When one has found $P = \{(\underline{p'_j}, \underline{p_j})_{1 \leq j \leq \ell}\}$, such that $\underline{p'_j} = \underline{p_j}S$, for all $j, 1 \leq j \leq \ell$, then S is a solution of the following linear system of equations:

$$\begin{cases} J_{\underline{b}}(\underline{p_j}) = J_{\underline{a}}(\underline{p'_j})X^t, \text{ for all } j, 1 \leq j \leq \ell. \\ \underline{p'_j} = \underline{p_j}X, \text{ for all } j, 1 \leq j \leq \ell. \end{cases}$$

In other words, $n \cdot \ell$ linear equations, given by P, relating the components of S are transformed into $n \cdot \ell \cdot (u+1)$ linear equations in the components of S. Thus ℓ must be chosen such that $n \cdot \ell \cdot (u+1) = n^2$, i.e. $\ell \approx \left\lceil \frac{n}{u+1} \right\rceil$ in order to obtain in this way (and without using (7)), n^2 linear equations in the components of S. However, we point out that equations generated in this way are not necessarily linearly independent.

Finally, we can also use a structural property of PLE to decrease the minimal value of ℓ required. Indeed, according to corollary 3, we have for all $d, 1 \leq d \leq D$:

$$J_{\underline{b}^{(d)}}(\underline{p}) = J_{\underline{a}^{(d)}}(\underline{p}S)S^t, \text{ for all } \underline{p} \in \mathbb{F}_q^n.$$

Notice that this last equation also holds for $d = 0$, but does not permit to get linear equations. Therefore, if $P = \{(\underline{p'_j}, \underline{p_j})_{1 \leq j \leq \ell}\}$ is a set of vector such that $\underline{p'_j} = \underline{p_j}S$, for all $j, 1 \leq j \leq \ell$, then S is a solution of the following linear system of equations:

$$\begin{cases} J_{\underline{b}}(\underline{p_j}) = J_{\underline{a}}(\underline{p'_j})X^t, \text{ for all } j, 1 \leq j \leq \ell. \\ J_{\underline{b}^{(d)}}(\underline{p_j}) = J_{\underline{a}^{(d)}}(\underline{p'_j})S^t, \text{ for all } d, 1 \leq d \leq D \text{ and for all } j, 1 \leq j \leq \ell. \\ \underline{p'_j} = \underline{p_j}X, \text{ for all } j, 1 \leq j \leq \ell. \end{cases} \quad (8)$$

In the sequel, $Sys(P)$ shall denote the linear system of equations obtained from (7) and (8).

We are now ready to present the PLE algorithm.

The algorithm

For a given $\ell \geq 1$, we select ℓ distinct (non-zero) vectors $\underline{p_1}, \ldots, \underline{p_\ell}$ and perform a so-called selective exhaustive search, which is detailed after the description of the PLE algorithm, to recover the corresponding vectors $\underline{p'_1} = \underline{p_1}S, \ldots, \underline{p'_\ell} = \underline{p_\ell}S$. The aim of this selective exhaustive search is to minimize the cost of constructing a set $P = \{(\underline{p'_j}, \underline{p_j})_{1 \leq j \leq \ell}\}$ such that $\underline{p'_j} = \underline{p_j}S$, for all $j, 1 \leq j \leq \ell$. We then compute the solutions of $Sys\big((\underline{p'_j}, \underline{p_j})_{1 \leq j \leq \ell}\big)$, denoted by $Sol\big(Sys\big((\underline{p'_j}, \underline{p_j})_{1 \leq j \leq \ell}\big)\big)$ in our algorithm, and the number of solutions of this linear system of equations. If it has less than C solutions (C is a small constant given in input of the algorithm), we try to find a solution of this system which is at the same time a linear equivalence matrix. If such a matrix exists then we return it. Otherwise and if, after having tried all the possible vectors $\underline{p'_1}, \ldots, \underline{p'_\ell}$ corresponding to $\underline{p_1}, \ldots, \underline{p_\ell}$, we have not obtained a linear equivalence matrix, we increment ℓ by 1 and restart the PLE algorithm with this new value of ℓ.

In the PLE algorithm, we use an auxiliary function, which we call Order, taking as input $n+1$ pairs of sets of vectors and returning these sets sorted in decreasing order (with respect to the number of elements in these sets).

The PLE algorithm
Input: $(\underline{a},\underline{b}) \in \mathbb{F}_q[\underline{x}]^u \times \mathbb{F}_q[\underline{x}]^u$, $(\ell, C) \in \mathbb{N}^* \times \mathbb{N}^*$.
Output: $S \in GL_n(\mathbb{F}_q)$, such that $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$.
$Sol_0 \leftarrow Sol\big(Sys(\underline{0}_n, \underline{0}_n)\big)$
If $|Sol_0| \leq C$ **then**
  If $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in Sol_0$ **then** return S
EndIf
`Selective Exhaustive Search: towards finding suitable pairs` $(\underline{p'},\underline{p})$
$Aux \leftarrow \big(\big(Ker\big(J^t_{\underline{a}}(\underline{0}_n)\big), Ker\big(J^t_{\underline{b}}(\underline{0}_n)\big)\big), (Ker(Q_{a_1}), Ker(Q_{b_1})), \ldots, (Ker(Q_{a_n}), Ker(Q_{b_n}))\big)$
$\big((A_0,B_0),\ldots,(A_n,B_n)\big) \leftarrow \text{Order}\big(Aux\big)$ and $(A_{n+1},B_{n+1}) \leftarrow (\mathbb{F}_q^n, \mathbb{F}_q^n)$
Let k be the minimum index such that $|\cup_{j=0}^{k} A_j| \geq \ell$
For i **from** 1 **to** k **do**
  $B_i \leftarrow B_i \setminus \cup_{j=0}^{i-1} B_j$ and $A_i \leftarrow A_i \setminus \cup_{j=0}^{i-1} A_j$
EndFor
$k' \leftarrow \ell - \sum_{j=0}^{k-1} |A_i|$ and $Aux \leftarrow A_0^{|A_0|} \times A_1^{|A_1|} \times \cdots \times A_k^{k'}$
`Select` $|B_0|$ `vectors in` B_0, $|B_1|$ `vectors in` B_1, `...,` `and` k' `vectors in` B_k
Randomly choose $(p_1,\ldots,p_\ell) \in B_0^{|B_0|} \times B_1^{|B_1|} \times \cdots \times B_k^{k'}$
`Search of a linear equivalence matrix`
While $\underline{b}(\underline{x}) \neq \underline{a}(\underline{x}S)$ **or** $Aux \neq \emptyset$ **do**
  `Select` $|A_0|$ `vectors in` A_0, $|A_1|$ `vectors in` A_1,`...,` `and` k' `vectors in` A_k
  Randomly choose $(p'_1,\ldots,p'_\ell) \in A_0^{|A_0|} \times A_1^{|A_1|} \times \cdots \times A_k^{k'}$
  $P \leftarrow \{(\underline{p'_j},\underline{p_j})_{1\leq j\leq \ell}\}$ and $Aux \leftarrow Aux \setminus \{p'_1,\ldots,p'_\ell\}$
  $Sol_P \leftarrow Sol\big(Sys(P)\big)$
  If $|Sol_P \cap Sol_0| \leq C$ **then**
    If $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in Sol_P \cap Sol_0$ **then** return S
  EndIf
EndWhile

The Selective Exhaustive Search
Let the notations be as in the PLE algorithm. One can see at once that, for all $i, 0 \leq i \leq n+1$, we have $A_i = B_iS$. We stress that such a property also holds after the first for loop. Moreover at each iteration of the PLE algorithm, by the very definition of k, it holds that $|\cup_{j=0}^{k-1} A_j| < \ell$, and thus $k' > 0$. Moreover, we have that $\ell = k' + \sum_{j=0}^{k-1} |B_i| = k' + \sum_{j=0}^{k-1} |A_i|$, since $|A_i| = |B_i|$, for all $i, 0 \leq i \leq n+1$.

In order to recover ℓ pairs of vectors $(\underline{p'_j},\underline{p_j})_{1\leq j\leq \ell}$, such that $\underline{p'_j} = \underline{p_j}S$, for all $j, 1 \leq j \leq \ell$, we select $|B_0|$ vectors $\underline{p_1},\ldots,\underline{p}_{|B_0|} \in B_0$, and perform an exhaustive search over $A_0(= B_0S)$ to recover the corresponding vectors $\underline{p'_1} = \underline{p_1}S,\ldots,$ $\underline{p'}_{|B_0|} = \underline{p}_{|B_0|}S$. We complete these $|B_0|$ vectors by choosing $|B_1|$ new vectors $\underline{p}_{|B_0|+1},\cdots,\underline{p}_{|B_0|+|B_1|} \in B_1$. The corresponding vectors $\underline{p'}_{|B_0|+1} = \underline{p}_{|B_0|+1}S,\ldots,$ $\underline{p'}_{|B_0|+|B_1|} = \underline{p}_{|B_0|+B_1|}S$ are recovered by performing an exhaustive search over $A_1(= B_1S)$. Finally, we complete the $\sum_{j=0}^{k-1} |B_i|$ vectors already chosen by se-

lecting $k'(= \ell - \sum_{j=0}^{k-1} |B_i|)$ new vectors $\underline{p}_{\ell-k'}, \ldots, \underline{p}_\ell \in B_k$. The corresponding vectors $\underline{p'}_{\ell-k'} = \underline{p}_{\ell-k'}S, \ldots, \underline{p'}_\ell = \underline{p}_\ell S$ are recovered by performing an exhaustive search over $A_k(= B_k S)$. Since, by construction, $|A_0| \leq |A_1| \leq \cdots \leq |A_k|$, we minimize in this way the cost of an exhaustive search for recovering the vectors $\underline{p'}_1 = \underline{p}_1 S, \ldots, \underline{p'}_\ell = \underline{p}_\ell S$.

5.2 Complexity

Let $\ell^* \in \mathbb{N}$ be the minimum value for which PLE returns a solution, i.e. the minimum number of pairs in P, for which $Sys(P)$ has n^2 linearly independent equations. As explained in 5.1, $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, for some $S \in GL_n(\mathbb{F}_q)$, implies that the linear equivalence matrix S verifies the following linear equations:

$$\begin{cases} J_{\underline{b}}(\underline{0}_n) = J_{\underline{a}}(\underline{0}_n)S^t \\ S^t Q_{b_j}^{-1} Q_{b_i} = Q_{a_j}^{-1} Q_{a_i} S^t, \forall 1 \leq j \leq n, \text{ s.t. } Q_{b_j} \text{ is invertible and } \forall 1 \leq i \neq j \leq n \end{cases}$$

These equalities allow us to obtain say nb_0 linearly independent equations in the components of S. We would like to emphasize that these equations are obtained in polynomial-time. Thus, if $nb_0 = n^2$ then our algorithm recovers S in polynomial-time.

Otherwise, if $nb_0 < n^2$, we have to find $\ell^* \geq 1$ pairs of non-zero vectors $(\underline{p}, \underline{p'})$, such that $\underline{p'} = \underline{p}S$. The cost of recovering these ℓ^* additional pairs being bounded from above by $q^{n\ell^*}$, the complexity of the PLE algorithm is:

$$O(n^6 q^{n\ell^*}),$$

which is the cost of solving a linear system of n^2 unknowns times the cost of recovering ℓ^* suitable pairs of vectors.

It seems difficult to obtain a precise value of ℓ^*. Anyway, in practice it appears that it is on the order of $\left\lceil \frac{n}{u+1} \right\rceil$. Finally, we mention that in order to minimize the number of pairs ℓ^* which has to be recovered, we can exploit a powerful idea that we shall call *exponentiation process*. It will be described in an extended version of this paper.

5.3 Practical Behaviour

We conclude this paper by giving some experimental results obtained with the PLE algorithm. The instances $(\underline{a}, \underline{b})$ of PLE have been generated in the following way. The polynomials of $\underline{a}$ have been randomly chosen of degree 2 or 3 (or more precisely with terms of total degree at most 2 or 3). To construct the polynomials of $\underline{b}$, we have randomly chosen $S \in GL_n(\mathbb{F}_q)$ and computed $\underline{b}(\underline{x}) = \big(a_1(\underline{x}S), \cdots, a_u(\underline{x}S)\big)$. The PLE algorithm described in 5.1 has been implemented using Magma software [8]. We have chosen the constant C (given in input of the PLE algorithm) equals to 10000. The results, obtained on a standard PC, are quoted in the following table. We mention that the times given in this table are in fact average times, obtained with our algorithm, for solving 10 instances of PLE (with u, n and q given).

n	u	q	degree	Time	degree	Time
50	50	$\mathbb{F}_{257}$	2	$\approx$ 0.3 s.	3	$\approx$ 10 s.
50	45	$\mathbb{F}_{257}$	2	$\approx$ 10 min.	3	$\approx$ 6 h.
60	60	$\mathbb{F}_{11}$	2	$\approx$ 0.2 s.	3	$\approx$ 10 s.
60	55	$\mathbb{F}_{11}$	2	$\approx$ 2 min.	3	$\approx$ 1 h.
60	50	$\mathbb{F}_{11}$	2	$\approx$ 2 min.	3	$\approx$ 1 h.
70	70	$\mathbb{F}_2$	2	$\approx$ 10 s.	3	$\approx$ 5 min.
70	65	$\mathbb{F}_2$	2	$\approx$ 10 s.	3	$\approx$ 5 min.
70	60	$\mathbb{F}_2$	2	$\approx$ 9 s.	3	$\approx$ 5 min.
70	55	$\mathbb{F}_2$	2	$\approx$ 9 s.	3	$\approx$ 5 min.

We would like to emphasize that the algorithms described in [7] have also been tested on these instances. The results are not quoted since these algorithms do not terminate (in a reasonable time). Anyway, we mention that in [11], the algorithms of [7] have been compared with a restricted version of the PLE algorithm described here. These experiments have been done on smaller instances of PLE (in terms of u, n and q) than the ones quoted in the above table. It appears that the PLE algorithm is much more efficient than the algorithms of [7] (which perform better than the algorithm described in [3]). This is mainly due to the fact that we have replaced the computation of Gröbner Bases by a Gaussian elimination.

Interpretation of the results

We first mention that the case $u \approx n$ is the most interesting for cryptographic applications of PLE. Indeed, in this setting it is very likely that an instance admits a unique solution (*see* [7] for further details). Moreover, $J_{\underline{b}}(\underline{0_n}) = J_{\underline{a}}(\underline{0_n})S^t$, allows us to obtain $n * Rank(J_{\underline{a}}(\underline{e_0}))$ linearly independent equations in the components S. Since $u \approx n$, then $Rank(J_{\underline{a}}(\underline{0_n}))$ is also close to n. Therefore, even if S is not uniquely determined by these equations, it is then very likely that a very little partial knowledge of S allows us to obtain a linear system of equation with less than C solutions. Since C is very small, we can quickly find if one of these C solutions is at the same time a linear equivalence matrix. Typically, in our experiments it has been sufficient to recover one pair $(\underline{p'}, \underline{p})$ such that $\underline{p'} = \underline{p}S$, confirming, at least for these parameters, that ℓ^* is close to $\left\lceil \frac{n}{u+1} \right\rceil$. Note that this pair is recovered efficiently using our selective exhaustive search.

Let us now analyze our results.

When $u = n$, and for $p = 257$ (resp. $p = 11$) the matrix $J_{\underline{a}}(\underline{0_n})$ was always invertible (in the ten instances generated). In this case, the solution is simply obtained by computing the transpose of $J_{\underline{a}}(\underline{0_n})^{-1} J_{\underline{b}}(\underline{0_n})$. For $p = 2$, it was not the case and we had to find only one pair $(\underline{p'}, \underline{p})$ such that $\underline{p'} = \underline{p}S$ in order to solve PLE. For this reason, our algorithm for $u = n$ is faster for $p = 257$ (resp. $p = 11$) than for $p = 2$. This result is in fact not surprising since the probability that a matrix $M \in \mathcal{M}_{n,n}(\mathbb{F}_q)$ is invertible is larger in $\mathbb{F}_{257}$ (resp. $\mathbb{F}_{11}$) than in $\mathbb{F}_2$. For instances $(\underline{a}, \underline{b})$ of PLE of degree 2, we can efficiently check if S is

indeed a linear equivalence matrix between $(\underline{a}, \underline{b})$. Let $A, B \in \mathcal{M}_{n,u}(\mathbb{F}_q)$ such that $\underline{a}^{(1)}(\underline{x}) = \underline{x}A$ and $\underline{b}^{(1)}(\underline{x}) = \underline{x}B$. Moreover, let Q_{a_i}, Q_{b_i} be, for all $i, 1 \leq i \leq u$, the unique matrices such that $a_i^{(2)}(\underline{x}) = \underline{x}Q_{a_i}\underline{x}^t$ and $b_i^{(2)}(\underline{x}) = \underline{x}Q_{b_i}\underline{x}^t$. According to property 2, we have $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$ iff $B = SA$ and $Q_{b_i} = SQ_{a_i}S^t$, for all $i, 1 \leq i \leq u$. Therefore, to check whether $\underline{b}(\underline{x}) = \underline{a}(\underline{x}S)$, we just have to compute product of matrices and compare these matrices. For an instance $(\underline{a}, \underline{b})$ of degree 3, such a manipulation is possible only for the homogeneous components of degree 1, and 2. But, in order to check whether $\underline{b}^{(3)}(\underline{x}) = \underline{a}^{(3)}(\underline{x}S)$ or not, we have to compute formally the polynomials $\underline{a}^{(3)}(\underline{x}S)$, which is much more costly than computing product of matrices (explaining the significant difference of results between instances of degree 2 and 3).

6 Conclusion

We have proved in this paper that IP1S and PLE are equivalent. Moreover, using a differential approach of PLE, we have presented a fast algorithm for solving PLE (and consequently also IP1S). It appears that, with the parameters proposed in [9], schemes based on IP1S are far from achieving the security level required for cryptographic applications. We recall that, initially, the security level of schemes based on IP1S has been estimated to $q^{\sqrt{2}n^{3/2}}$[10].

Acknowledgements

I would like to thank F.Levy-dit-Vehel for helpful discussions related to this paper.

References

1. N. Courtois, L. Goubin, and J. Patarin: Improved Algorithms for Isomorphism of Polynomials. *Advances in Cryptology - EUROCRYPT '98, Lecture Notes in Computer Science,* vol. 1403, Springer-Verlag, pp. 84–200, 1998.
2. N. Courtois, L. Goubin, and J. Patarin: Improved Algorithms for Isomorphism of Polynomials - Extended Version. Available from `www.minrank.org`.
3. W.Geiselmann, W.Meier, and R.Steinwandt: An Attack on the Isomorphisms of Polynomials Problem with One Secret. Int. Journal of Information Security, Vol. 2(1): pp. 59–64, 2003.
4. O. Goldreich, S. Micali, and A. Wigderson: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. Journal of the ACM, Vol. 38(3) pp. 690–728, 1991.
5. M. R. Garey, and D. B. Johnson: Computers and Intractability. A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979.
6. S. Goldwasser, S. Micali, and C. Rackoff: The Knowledge Complexity of Interactive Proof Systems. SIAM J. on Computing, Vol. 18, pp. 186–208, 1989.
7. F. Levy-dit-Vehel, and L. Perret: Polynomial equivalence problems and applications to multivariate cryptosystems. *Progress in Cryptology - INDOCRYPT 2003, Lecture Notes in Computer Science,* vol. 2904, pp. 235–251, 2003.

8. http://magma.maths.usyd.edu.au/magma/
9. J. Patarin: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms. *Advances in Cryptology - EUROCRYPT '96, Lecture Notes in Computer Science,* vol. 1070, Springer-Verlag, pp. 33–48, 1996.
10. J. Patarin:Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms - Extended Version. Available from `www.minrank.org/hfe/`.
11. L. Perret, and A. Bayad: A differential approach to a polynomial equivalence problem, in Proceedings of ISIT 2004, extended abstract, pp. 142, 2004.

Partial Key Exposure Attacks on RSA Up to Full Size Exponents

Matthias Ernst[1], Ellen Jochemsz[2,⋆],
Alexander May[1,⋆], and Benne de Weger[2,⋆]

[1] Faculty of Computer Science, Electrical Engineering and Mathematics,
University of Paderborn, 33102 Paderborn, Germany
{alder, alexx}@uni-paderborn.de
[2] Department of Mathematics and Computer Science,
Eindhoven University of Technology, Eindhoven, The Netherlands
{e.jochemsz, b.m.m.d.weger}@tue.nl

Abstract. We present several attacks on RSA that factor the modulus in polynomial time under the condition that a fraction of the most significant bits or least significant bits of the private exponent is available to the attacker. Our new attacks on RSA are the first attacks of this type that work up to full size public or private exponent.

Keywords: RSA, cryptanalysis, partial key exposure, lattice reduction, Coppersmith's method.

1 Introduction

There have been a number of attacks on RSA given a portion of the private key. These attacks are so-called partial key exposure attacks, where an attacker has some knowledge of the bits of the private key and uses it to break the system. The results are of practical interest, since implementations may leak bits of the private key, e.g. via side channel attacks.

In 1998, Boneh, Durfee and Frankel presented several partial key exposure attacks on RSA in [2]. Some of these attacks require knowledge of the least significant bits (LSBs) of the private exponent, others of the most significant bits (MSBs). Additionally, in their attacks, the public exponent must be relatively small. Wiener's attack [12] and the improvement by Boneh and Durfee [1] can be seen as partial key exposure attacks where the most significant bits of the private exponent are known to be equal to zero.

In [2] the question is posed whether there exist partial key exposure attacks on RSA that work for public exponents larger than the square root of the modulus.

⋆ The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 371–386, 2005.

In 2003, Blömer and May [3] described a number of attacks that do allow larger public exponents, but not yet to the full size of the modulus. In this paper we present attacks for full size public exponent that work up to full size private exponent. Additionally, we present a new attack for full size private exponent that works up to full size public exponent.

Our attacks use Coppersmith's ideas of finding small roots of polynomials [4]. We look at variations on the RSA key equation over the integers, using Coppersmith's method of finding small integer roots, reformulated by Coron [5].

Our new results on known MSBs of d for small private exponent d and full size public exponent e are summarized in the following theorem.

Theorem 1 (MSB small d). *Under a common heuristic assumption concerning resultants, for every $\epsilon > 0$, there exists n_0 such that for every $n > n_0$, the following holds:*
Let $N = pq$ be an n-bit RSA-modulus, and p, q primes of bitsize $\frac{n}{2}$. Let $0 < \delta < \beta < 1$. Furthermore, let e, d satisfy $ed \equiv 1 \bmod \phi(N)$ with bitsize$(e) = n$ and bitsize$(d) = \beta n$. Given the $(\beta - \delta)n$ MSBs of d, N can be factored in polynomial time if:

- $\delta \leq \frac{5}{6} - \frac{1}{3}\sqrt{1 + 6\beta} - \epsilon$ *(Section 4.1.1), or*
- $\delta \leq \frac{3}{16} - \epsilon$ *and* $\beta \leq \frac{11}{16}$ *(Section 4.1.2), or*
- $\delta \leq \frac{1}{3} + \frac{1}{3}\beta - \frac{1}{3}\sqrt{4\beta^2 + 2\beta - 2} - \epsilon$ *and* $\beta \geq \frac{11}{16}$ *(Section 4.1.2).*

In the case of known MSBs for full size d and small e, we find an improvement of known results by [2] and [3] for $e \in [N^{\frac{1}{2}}, N]$. Our result is stated in the theorem below.

Theorem 2 (MSB small e). *Under a common heuristic assumption concerning resultants, for every $\epsilon > 0$, there exists n_0 such that for every $n > n_0$, the following holds:*
Let $N = pq$ be an n-bit RSA-modulus, and p, q primes of bitsize $\frac{n}{2}$. Let $0 < \delta < \frac{1}{2} < \alpha < 1$. Let e, d satisfy $ed \equiv 1 \bmod \phi(N)$, such that bitsize$(d) = n$ and bitsize$(e) = \alpha n$.
Given the $(1 - \delta)n$ MSBs of d, N can be factored in polynomial time if:

- $\delta \leq \frac{1}{3} + \frac{1}{3}\alpha - \frac{1}{3}\sqrt{4\alpha^2 + 2\alpha - 2} - \epsilon$ *(Section 4.2).*

In Fig. 1 and 2 we illustrate our results on known MSBs of d. In Fig. 1, the fraction of bits required for an attack is plotted as a function of the size of d. It shows the parts of the key space that are insecure by the attacks in Section 4.1, and by the results of [12] and [1]. Fig. 2 is a picture of the relation between the fraction of bits of d required for an attack and the size of e, showing the results of [2], [3], and Section 4.2.

Note that our attacks for known MSBs have natural starting and ending points. One MSB attack on small d coincides with the bound $d \leq N^{0.284}$ from [1], the other runs up to the situation where d is of full size and is fully known. This links our results to that of May [9], proving a deterministic polynomial time

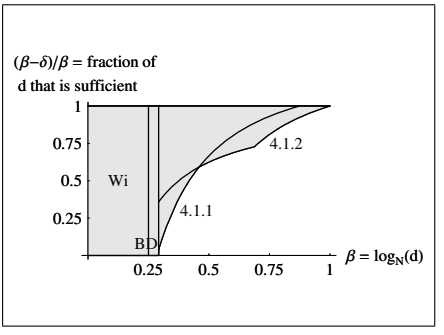

Fig. 1. MSB attacks for small d

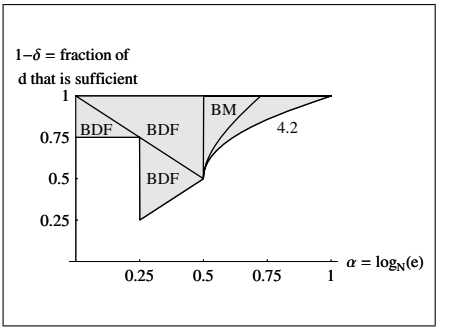

Fig. 2. MSB attacks for small e

equivalence between factoring and full knowledge of d. Our MSB attack on small e is a natural extension of the results of [2] and [3].

Our new result on known LSBs for relatively small d and full size e is as follows.

Theorem 3 (LSB small d). *Under a common heuristic assumption concerning resultants, for every $\epsilon > 0$, there exists n_0 such that for every $n > n_0$, the following holds:*
Let $N = pq$ be an n-bit RSA-modulus, and p, q primes of bitsize $\frac{n}{2}$. Let $0 < \delta < \beta < 1$. Furthermore, let e, d satisfy $ed \equiv 1 \bmod \phi(N)$ with bitsize$(e) = n$ and bitsize$(d) = \beta n$. Given the $(\beta - \delta)n$ LSBs of d, N can be factored in polynomial time when:

- $\delta < \frac{5}{6} - \frac{1}{3}\sqrt{1 + 6\beta} - \epsilon$ *(Section 4.3).*

Fig. 3 illustrates our result on known LSBs. The fraction of bits required for an attack is plotted as a function of the size of d. Fig. 4 is a picture of the relation between the fraction of bits required for an attack, and the size of e, showing the work of [2] and [3]. Analysis of our LSB method in the case where e is small results in a bound equivalent to the best result of [3].

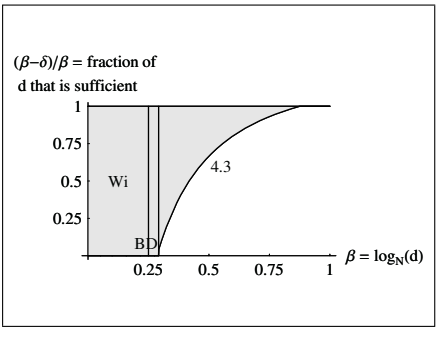

Fig. 3. LSB attacks for small d

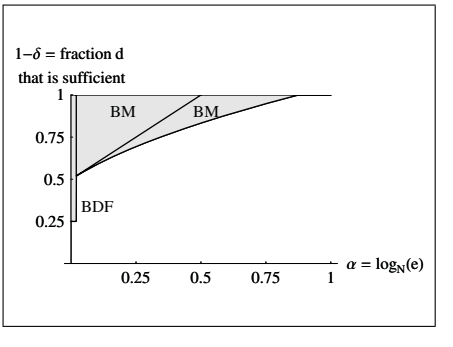

Fig. 4. LSB attacks for small e

Again, notice that the starting point of our new LSB attack for small d coincides with the bound $d \leq N^{0.284}$ from [1].

The bounds of Section 4.1.1 and 4.3 show a symmetry in the outcomes of the MSB and LSB situations for small d. Likewise, the bound of Section 4.2 and the second bound of Section 4.1.2 show a symmetry in the outcomes of the MSB cases for small d and small e. This is a result of the general character of our method. Instead of studying only special scenarios, we analyze the underlying polynomials in a general framework, which also makes it possible to study numerous old and new cryptanalytic situations, on which we shall comment in Section 4.4.

Our results can be viewed as evidence that side channel attacks are even more dangerous to RSA than we already knew. In essence, we show that there exist partial key exposure attacks up to full size exponents, hence if either e or d is chosen to be significantly smaller than $\phi(N)$, the system is vulnerable to this type of attacks. This can be understood as a warning to crypto-designers to choose both private and public exponent at random, or take sufficient countermeasures to prevent private key bits from leaking.

This paper is organized as follows. In Section 2, we describe the typical RSA setting and show how we derive polynomials with small roots from the RSA key equation when MSBs or LSBs of the private exponent d are known. In Section 3, we give an overview of the tools we use to find the small roots of these polynomials. In Section 4 we give the description of our attacks, proving the results of Theorem 1, 2, and 3. In Section 5, experimental results are provided.

2 Looking at the RSA Key Equation

Let p, q, N, d, e be as usual, i.e. p and q are distinct primes, $N = pq$ is taken as modulus, and the encryption exponent e and decryption exponent d satisfy $ed \equiv 1 \bmod \phi(N)$. For all of the attacks in this paper, we assume that p and q have the same bitsize, thus $p + q < 3\sqrt{N}$. Let $k \in \mathbb{Z}$ be defined by the RSA key equation

$$ed - 1 = k\phi(N), \quad \text{where } \phi(N) = (p-1)(q-1) = N - (p + q - 1).$$

In our scenario, we assume one of the exponents e and d is chosen to be small and the other one is of full size. We will first focus on the case where d is small. Therefore, we place no restrictions on e except $e < \phi(N)$. It follows that $k < \frac{ed}{\phi(N)} < d$.

When MSBs of d are known, we write $d = \tilde{d} + d_0$, where $\tilde{d}$ (representing the most significant bits of d) is known to the attacker, but d_0 (representing the least significant bits of d) is not. To make this precise, let β and δ be parameters such that $d \leq N^{\beta}$, and $|d_0| = |d - \tilde{d}| \leq N^{\delta}$.

For the MSB case, we can thus rewrite the RSA key equation into

$$e(\tilde{d} + d_0) - 1 = k(N - (p + q - 1)).$$

Hence, the polynomial

$$f_{MSB1}(x, y, z) = ex - Ny + yz + R, \text{ where } R = e\tilde{d} - 1,$$

has a root $(x_0, y_0, z_0) = (d_0, k, p+q-1)$. Let $X := N^\delta$, $Y := N^\beta$, and $Z := 3N^{\frac{1}{2}}$. Then the root is 'small' since $|x_0| < X$, $|y_0| < Y$, and $|z_0| < Z$.

The attacker can also compute $\tilde{k} = \frac{e\tilde{d}-1}{N}$ as an approximation to k, and set $k_0 = k - \tilde{k}$ as the unknown part of k. It can be shown (as was done in [3]) that $|k_0| < \frac{e}{\phi(N)}(N^\delta + 3N^{\beta-\frac{1}{2}})$, so in our case we have $|k_0| < 4N^\gamma$, where $\gamma = \max\{\delta, \beta - \frac{1}{2}\}$. When we substitute the knowledge of the MSBs of k into the RSA key equation, we obtain

$$e(\tilde{d} + d_0) - 1 = (\tilde{k} + k_0)(N - (p + q - 1)).$$

Hence,

$$f_{MSB2}(x, y, z) = ex - Ny + yz + \tilde{k}z + R, \text{ with } R = e\tilde{d} - 1 - \tilde{k}N,$$

has a root $(x_0, y_0, z_0) = (d_0, k_0, p + q - 1)$. With $X := N^\delta$, $Y := 4N^\gamma$, and $Z := 3N^{\frac{1}{2}}$, we have $|x_0| < X$, $|y_0| < Y$, and $|z_0| < Z$.

When LSBs of d are known, the attacker knows $\bar{d} \equiv d \bmod M$ for some M, and we write $d = \bar{d} + d_1 M$, where $\bar{d}$ and M are known, and d_1 is not. We assume that $d \leq N^\beta$, and $d_1 \leq N^\delta$. We have no approximation of k in this case, so we rewrite the RSA key equation as

$$e(d_1 M + \bar{d}) - 1 = k(N - (p + q - 1)).$$

Thus,

$$f_{LSB}(x, y, z) = eMx - Ny + yz + R, \text{ with } R = e\bar{d} - 1,$$

has a root $(x_0, y_0, z_0) = (d_1, k, p + q - 1)$. Using $X := N^\delta$, $Y := N^\beta$, and $Z := 3N^{\frac{1}{2}}$, we have $|x_0| < X$, $|y_0| < Y$, and $|z_0| < Z$.

3 Finding Small Roots

We have seen that in several cases, we can obtain d, k and $p+q-1$ when we can find a small root of a certain trivariate polynomial. In this section, we describe some tools that we use to solve this problem of finding small roots. For a polynomial $h(x, y, z) = \sum_{i,j,k} h_{ijk} x^i y^j z^k$, we define $||h(x, y, z)||^2 := \sum_{i,j,k} |h_{ijk}|^2$ and $||h(x, y, z)||_\infty := \max_{i,j,k} |h_{ijk}|$.

In [4], Coppersmith describes rigorous techniques to find small integer roots of polynomials in a single variable modulo n, and of polynomials in two variables over the integers. The methods extend to more variables, making them heuristical. Howgrave-Graham reformulated Coppersmith's ideas of finding modular roots in [6], of which we use the following lemma.

Lemma 1 (Howgrave-Graham). *Let $h(x, y, z) \in \mathbb{Z}[x, y, z]$ be a polynomial which is a sum of at most ω monomials. Suppose that $h(x_0, y_0, z_0) \equiv 0 \bmod n$ for some $|x_0| < X$, $|y_0| < Y$, $|z_0| < Z$, and $||h(xX, yY, zZ)|| < \frac{n}{\sqrt{\omega}}$. Then $h(x_0, y_0, z_0) = 0$ holds over the integers.*

Howgrave-Graham's lemma is usually combined with LLL reduction of lattice bases [7].

Fact 1 (LLL). *Let L be a lattice of dimension ω. In polynomial time, the LLL-algorithm outputs two reduced basis vectors v_1 and v_2, that satisfy*

$$||v_1|| \leq ||v_2|| \leq 2^{\frac{\omega}{4}} \det(L)^{\frac{1}{\omega-1}}.$$

Thus, the condition $2^{\frac{\omega}{4}} \det(L)^{\frac{1}{\omega-1}} < \frac{n}{\sqrt{\omega}}$ implies that polynomials corresponding to the two shortest reduced basis vectors match Howgrave-Graham's bound. This condition reduces to $\det(L) \leq (2^{\frac{-\omega}{4}} \frac{1}{\sqrt{\omega}})^{\omega-1} n^{\omega-1}$. In practice, we ignore terms that do not depend on n, and check only if $\det(L) \leq n^{\omega-1}$.

Coppersmith's technique of finding small roots of polynomials over the integers has so far been less applied in cryptanalysis methods. Recently, Coron [5] reformulated this technique analogous to Howgrave-Graham. Essentially, Coron picks a 'suitable' integer n and transfers the situation into finding a small root modulo n, thereby applying Howgrave-Graham's lemma. In the following sections, we will study the polynomials f_{MSB1}, f_{MSB2}, and f_{LSB} to find their small roots over the integers, analogous to Coron.

4 Description of the Attacks

4.1 Known MSBs and Small d

4.1.1 Attack Using f_{MSB1}

We will now describe a method that finds a small root of f_{MSB1} over the integers, and prove the first result of Theorem 1, namely that we have a polynomial time MSB attack when

$$\delta \leq \frac{5}{6} - \frac{1}{3}\sqrt{1+6\beta} - \epsilon.$$

Recalling the situation where we do not use an approximation of k, we want to find a small root (x_0, y_0, z_0) of the polynomial $f_{MSB1}(x, y, z) = ex - Ny + yz + R$.

Our first observation is that f_{MSB1} is irreducible over the integers. Thus, if we could construct two polynomials f_1, f_2 with the same root (x_0, y_0, z_0) which are not multiples of f_{MSB1}, then they do not share a common divisor with f_{MSB1}. Hence, the polynomials $p_1(y, z) = \mathrm{Res}_x(f_{MSB1}, f_1)$ and $p_2(y, z) = \mathrm{Res}_x(f_{MSB1}, f_2)$ cannot be the zero polynomials. Under the heuristic that the resultant $\mathrm{Res}_y(p_1, p_2)$ does not vanish, we obtain $z_0 = p + q - 1$ from a linear factor $(z - z_0)$ in $\mathrm{Res}_y(p_1, p_2)$, which gives us the factorization of N. All attacks in this paper have a similar heuristic concerning resultants, common in cryptanalysis using multivariate polynomials. Therefore we will use the following assumption.

Assumption 1. *The resultant computations for the polynomials in this paper yield non-zero polynomials.*

We will comment on how this assumption holds in practice in Appendix D, where we also provide experimental results.

Now let us find conditions under which we can construct f_1 and f_2 as defined above. Let X, Y, Z be upper bounds for x_0, y_0, z_0, respectively. We fix an integer m depending on $\frac{1}{\epsilon}$, and a parameter t, that we will optimize later in terms of m. We define $W = ||f_{MSB1}(xX, yY, zZ)||_\infty$ and $n = (XY)^m Z^{m+t} W$.

First, in order to work with a polynomial with constant term 1, we define

$$f(x, y, z) \equiv R^{-1} f_{MSB1}(x, y, z) \bmod n \equiv 1 + ax + by + cyz.$$

Let us look at the following collection of polynomials, the so-called shifts:

$$\begin{aligned} g_{ijk}(x, y, z) &= x^i y^j z^k f(x, y, z) X^{m-i} Y^{m-j} Z^{m+t-k}, \\ &\quad \text{for } i = 0, \ldots, m \ ; \ j = 0, \ldots, m - i \ ; \ k = 0, \ldots, j, \\ h_{ijk}(x, y, z) &= x^i y^j z^k f(x, y, z) X^{m-i} Y^{m-j} Z^{m+t-k}, \\ &\quad \text{for } i = 0, \ldots, m \ ; \ j = 0, \ldots, m - i \ ; \ k = j + 1, \ldots, j + t. \end{aligned}$$

In addition to the polynomials g and h, we also use the polynomials

$$\begin{aligned} g'_{ijk}(x, y, z) &= n x^i y^j z^k \text{ for } i = 0, \ldots, m + 1; j = m + 1 - i; k = 0, \ldots, j, \\ h'_{ijk}(x, y, z) &= n x^i y^j z^k \text{ for } i = 0, \ldots, m + 1; j = m + 1 - i; k = j + 1, \ldots, j + t. \end{aligned}$$

Let us give an intuition of the construction of our collection of polynomials. When $m = t = 1$, the polynomials g are constructed by multiplying f by its monomials 1, x, y, yz and constants. In this way, all the monomials of f^2 appear. In general, the polynomials g are constructed by multiplying f by the monomials of f^m, thereby creating the monomials of f^{m+1}. Additionally, we add the z-shifts h. In our example, the z-shifts are constructed by multiplying f by the terms z, xz, and yz^2 and constants. The terms z, xz, and yz^2 are the multiplications of z and the original monomials, without yz since this shift was already in g. The auxiliary polynomials g', h' contain the monomials in g and h that were not used for shifts.

Obviously, g, h, g', and h' all have the root (x_0, y_0, z_0) modulo n: g and h have f as a factor, and g' and h' are multiples of n. Let f_1 and f_2 be linear combinations of these polynomials. According to Howgrave-Graham's lemma, if $||f_1(xX, yY, zZ)||$ and $||f_2(xX, yY, zZ)||$ are smaller than $\frac{n}{\sqrt{\omega}}$, then f_1 and f_2 both have the root (x_0, y_0, z_0) over the integers.

Moreover, we want to ensure that $f_1(xX, yY, zZ)$ and $f_2(xX, yY, zZ)$ are not multiples of $f_{MSB1}(xX, yY, zZ)$, which implies that f_1, f_2 are not multiples of f. By construction, each of our polynomials $g_{ijk}(xX, yY, zZ)$, $h_{ijk}(xX, yY, zZ)$, $g'_{ijk}(xX, yY, zZ)$, $h'_{ijk}(xX, yY, zZ)$ is divisible by $(XY)^m Z^{m+t}$. So $f_1(xX, yY, zZ)$ and $f_2(xX, yY, zZ)$ must be divisible by this term. According to a lemma of Coron [5–Lemma 3], for any multiple $h(xX, yY, zZ)$ of $f_{MSB1}(xX, yY, zZ)$ it holds that

$$||h(xX, yY, zZ)|| \geq 2^{-(\rho+1)^2} (XY)^m Z^{m+t} \cdot ||f_{MSB1}(xX, yY, zZ)||_\infty = 2^{-(\rho+1)^2} n,$$

where ρ is the maximum degree of the polynomials h and f_{MSB1} in each variable separately. If we let terms that do not depend on n contribute to ϵ, we find that a linear combination with norm smaller than n cannot be a multiple of f_{MSB1}, and must satisfy Howgrave-Graham's bound.

We build a lattice L using as a basis the coefficient vectors of $g_{ijk}(xX, yY, zZ)$, $h_{ijk}(xX, yY, zZ)$, $g'_{ijk}(xX, yY, zZ)$ and $h'_{ijk}(xX, yY, zZ)$. We order the vectors such that the matrix is triangular, and the diagonal entries of g and h are equal to $(XY)^m Z^{m+t}$. For $m = t = 1$, after dividing out XYZ^2 for simplicity, the coefficient matrix is the following (the rows correspond to the coefficient vectors of h, g, h', and g', respectively).

$$
\begin{array}{c|cccccc|cccccccccc}
 & z & xz & yz^2 & 1 & x & y & yz & x^2z & xyz^2 & y^2z^3 & x^2 & xy & y^2 & y^2z & xyz & y^2z^2 \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\hline
 & 1 & aX & cYZ & & & & bY & & & & & & & & & \\
 & & 1 & & & & & & aX & cYZ & & & & & & bY & \\
 & & & 1 & & & & & & aX & cYZ & & & & & & bY \\
\hline
 & & & & 1 & aX & bY & cYZ & & & & & & & & & \\
 & & & & & 1 & & & & & & aX & bY & & & cYZ & \\
 & & & & & & 1 & & & & & & aX & bY & cYZ & & \\
 & & & & & & & 1 & & & & & & & bY & aX & cYZ \\
\hline
 & & & & & & & & WX^2Z & & & & & & & & \\
 & & & & & & & & & WXYZ^2 & & & & & & & \\
 & & & & & & & & & & WY^2Z^3 & & & & & & \\
\hline
 & & & & & & & & & & & WX^2 & & & & & \\
 & & & & & & & & & & & & WXY & & & & \\
 & & & & & & & & & & & & & WY^2 & & & \\
 & & & & & & & & & & & & & & WY^2Z & & \\
 & & & & & & & & & & & & & & & WXYZ & \\
 & & & & & & & & & & & & & & & & WY^2Z^2
\end{array}
$$

In general, the computations in Appendix A show that for $t = \tau m$, if

$$X^{1+3\tau} Y^{2+3\tau} Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau},$$

we find polynomials f_1 and f_2 that satisfy the Howgrave-Graham bound. Thus, they have the root (x_0, y_0, z_0) over the integers and are not multiples of f. Under Assumption 1, the resultant method will reveal the integer root (x_0, y_0, z_0). Note that the bound can be applied on any irreducible polynomial with the monomials 1, x, y, and yz.

In our case, $X = N^\delta$, $Y = N^\beta$, $Z = 3N^{\frac{1}{2}}$ and $W = \max\{eX, NY, YZ, R\} \geq NY = N^{1+\beta}$. We find an optimal value $\tau = \frac{1}{2} - \delta$, which implies $\delta \leq \frac{5}{6} - \frac{1}{3}\sqrt{1+6\beta}$. Thereby, we have derived the first result of Theorem 1.

4.1.2 Attack Using f_{MSB2}

We will now show how to obtain the second and third result mentioned in Theorem 1, namely that we have a polynomial time MSB attack whenever

$$\delta \leq \frac{3}{16} - \epsilon \text{ and } \beta \leq \frac{11}{16}, \quad \text{or} \quad \delta \leq \frac{1}{3} + \frac{1}{3}\beta - \frac{1}{3}\sqrt{4\beta^2 + 2\beta - 2} - \epsilon \text{ and } \beta \geq \frac{11}{16}.$$

For the situation where we use information on MSBs of d to get an approximation $\tilde{k}$ of k, we want to find a small root (x_0, y_0, z_0) of the polynomial $f_{MSB2}(x, y, z) = ex - Ny + yz + \tilde{k}z + R$.

We fix an integer m depending on $\frac{1}{\epsilon}$, a parameter t that we optimize later, and put $W = ||f_{MSB2}(xX, yY, zZ)||_\infty$, and $n = X^m Y^{m+t} Z^m W$. We compute $f \equiv R^{-1} f_{MSB2} \bmod n \equiv 1 + ax + by + cyz + dz$, and define the collection of polynomials

$$\begin{aligned}
g_{ijk}(x,y,z) &= x^i y^j z^k f(x,y,z) X^{m-i} Y^{m+t-j} Z^{m-k}, \\
&\quad \text{for } i = 0,\ldots,m \ ;\ j = 0,\ldots,m-i \ ;\ k = 0,\ldots,m-i, \\
h_{ijk}(x,y,z) &= x^i y^j z^k f(x,y,z) X^{m-i} Y^{m+t-j} Z^{m-k}, \\
&\quad \text{for } i = 0,\ldots,m \ ;\ j = m-i+1,\ldots,m-i+t \ ;\ k = 0,\ldots,m-i, \\
g'_{ijk}(x,y,z) &= n x^i y^j z^k, \\
&\quad \text{for } i = 0,\ldots,m+1 \ ;\ j = 0,\ldots,m+t+1-i \ ;\ k = m+1-i, \\
h'_{ijk}(x,y,z) &= n x^i y^j z^k, \\
&\quad \text{for } i = 0,\ldots,m \ ;\ j = m+t+1-i \ ;\ k = 0,\ldots,m-i.
\end{aligned}$$

As before, the polynomials g are constructed by shifting f with every monomial of f^m. The polynomials h represent extra y-shifts, the shifts used are y^l times the monomials of f^m, for $l = 1,\ldots,t$ (excluding shifts that were already in g). The auxiliary polynomials g' and h' contain the monomials of g and h that were not used for the shifts.

We build a lattice L using as a basis the coefficient vectors of $g_{ijk}(xX, yY, zZ)$, $h_{ijk}(xX, yY, zZ)$, $g'_{ijk}(xX, yY, zZ)$, and $h'_{ijk}(xX, yY, zZ)$, where we order the vectors such that the corresponding lattice basis is triangular, and the diagonal entries of g and h are equal to $X^m Y^{m+t} Z^m$.

The computations in Appendix B show for $t = \tau m$, that when

$$X^{2+3\tau} Y^{3+6\tau+3\tau^2} Z^{3+3\tau} \leq W^{2+3\tau}$$

holds, we can find two reduced basis vectors that satisfy the Howgrave-Graham bound. So under Assumption 1, we can find the factorization of N in polynomial time.

In our case, we have $X = N^\delta$, $Y = 4N^\gamma$, with $\gamma = \max\{\delta, \beta - \frac{1}{2}\}$, and $Z = 3N^{\frac{1}{2}}$. Also, $W = \max\{eX, NY, YZ, \tilde{k}Z, R\} \geq NY = 4N^{1+\gamma}$. The optimal value $\tau = \frac{\frac{1}{2}-\delta-\gamma}{2\gamma}$ leads to the condition $\delta \leq \frac{1}{3}\gamma + \frac{1}{2} - \frac{1}{3}\sqrt{4\gamma^2 + 6\gamma}$. If $\gamma = \delta$, this implies $\delta \leq \frac{3}{16}$, valid for $\beta \leq \frac{11}{16}$. If $\gamma = \beta - \frac{1}{2}$, we get $\delta \leq \frac{1}{3} + \frac{1}{3}\beta - \frac{1}{3}\sqrt{4\beta^2 + 2\beta - 2}$, valid for $\beta \geq \frac{11}{16}$.

This concludes the proof of Theorem 1.

4.2 Known MSBs and Small e

In practice, one often chooses the public exponent e to be small. Therefore, we now let $e = N^\alpha$ and $d < \phi(N)$. Note that we are able to use the same polynomials f_{MSB1} and f_{MSB2}, when we make some changes in the size of the parameters. The best result in this situation, as mentioned in Theorem 2, is that we obtain a polynomial time MSB attack whenever

$$\delta \leq \frac{1}{3} + \frac{1}{3}\alpha - \frac{1}{3}\sqrt{4\alpha^2 + 2\alpha - 2} \ \text{ for } \alpha > \frac{1}{2}.$$

We can again use $f_{MSB1}(x,y,z) = ex - Ny + yz + R$, now with $|d_0| < X = N^{\delta}$, $|k| < Y = N^{\alpha}$ and $|p+q-1| < Z = 3N^{\frac{1}{2}}$. Using $W = N^{1+\alpha}$, as in Section 4.1.1 we find $\delta \leq \frac{5}{6} - \frac{1}{3}\sqrt{1+6\alpha}$. This result only holds for $\alpha > \frac{1}{2}$. In the case $\alpha < \frac{1}{2}$, from [2–Theorem 4.1], we can assume that k is known, and the polynomial to be analyzed becomes bivariate. Since our attack using f_{MSB1} obtains a worse bound than the one using f_{MSB2}, it is not mentioned in Theorem 2.

When we use partial information on k, where k is partly unknown (so $\alpha > \frac{1}{2}$), we can use $f_{MSB2}(x,y,z) = ex - Ny + yz + \tilde{k}z + R$. We have $|d_0| < X = N^{\delta}$, $|k_0| < Y = 4N^{\gamma}$, with $\gamma = \max\{\alpha + \delta - 1, \alpha - \frac{1}{2}\}$, and $|p+q-1| < Z = 3N^{\frac{1}{2}}$. Using $W = N^{1+\gamma}$, we get the same condition as in the previous paragraph, namely $\delta \leq \frac{1}{3}\gamma + \frac{1}{2} - \frac{1}{3}\sqrt{4\gamma^2 + 6\gamma}$, that we analyze for two possibilities for γ.

If we substitute $\gamma = \alpha + \delta - 1$ (in other words, we assume $\delta > \frac{1}{2}$), we obtain the condition $\delta < \frac{3+4\alpha-4\alpha^2}{16\alpha}$. However, for $\alpha > \frac{1}{2}$, $\delta < \frac{3+4\alpha-4\alpha^2}{16\alpha} < \frac{1}{2}$, so we get no result. If $\gamma = \alpha - \frac{1}{2}$, we find $\delta \leq \frac{1}{3} + \frac{1}{3}\alpha - \frac{1}{3}\sqrt{4\alpha^2 + 2\alpha - 2}$. This concludes the proof of Theorem 2.

4.3 Known LSBs and Small *d*

In this section, we will show how to obtain the result of Theorem 3, namely that we have a polynomial time LSB attack whenever

$$\delta \leq \frac{5}{6} - \frac{1}{3}\sqrt{1+6\beta} - \epsilon.$$

The polynomial $f_{LSB}(x,y,z) = eMx - Ny + yz + R$, where $R = e\bar{d} - 1$, has the same monomials as f_{MSB1}. So we can directly apply the analysis of Section 4.1.1. We use

$$X^{1+3\tau}Y^{2+3\tau}Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau},$$

on $X = N^{\delta}$, $Y = N^{\beta}$, $Z = 3N^{\frac{1}{2}}$ and $W = \max\{eMX, NY, YZ, R\} \geq NY = N^{1+\beta}$. This implies $\delta \leq \frac{5}{6} - \frac{1}{3}\sqrt{1+6\beta}$, which concludes the proof of Theorem 3.

If we adapt the LSB attack for the situation when e is not of full size, we get exactly the result from Blömer and May in [3–Section 6].

4.4 Other Applications of the General Method

We have already mentioned that the analysis of the approaches using f_{MSB1} and f_{MSB2} is general, in the sense that for every irreducible polynomial with the monomials $1, x, y, yz$, the inequality $X^{1+3\tau}Y^{2+3\tau}Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau}$ defines the condition for a successful (heuristic) attack. Also, for every irreducible polynomial with the monomials $1, x, y, yz, z$, the condition $X^{2+3\tau}Y^{3+6\tau+3\tau^2}Z^{3+3\tau} \leq W^{2+3\tau}$ implies a successful (heuristic) attack. As a consequence, many known attacks on RSA are special cases of our general framework.

As could be seen in Fig. 1, and 3, one MSB attack and one LSB attacks for small d coincide with the bound $d < N^{0.284}$ from [1]. This result can also be found

using our method by noticing that $f_{BD}(x, y, z) = ex - Ny + yz - 1$ with the root $(x_0, y_0, z_0) = (d, k, p+q-1)$ has the same monomials as f_{MSB1}. Therefore, we can substitute $X, Y = N^{\beta}$, $Z = 3N^{\frac{1}{2}}$ and $W = \max\{eX, NY, YZ\} \geq NY = N^{1+\beta}$ in $X^{1+3\tau}Y^{2+3\tau}Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau}$, which leads to $\beta \leq \frac{7}{6} - \frac{1}{3}\sqrt{7} \approx 0.284$. To get the improved Boneh-Durfee bound one has to use sublattices. We leave this for further research.

Fig. 1 also shows that the graph of our (asymptotic) attack of Section 4.1.2 goes up to $\frac{\beta-\delta}{\beta} = 1$, for the parameter choice $\beta = 1$. This links our result to a recent result of May [9], who proves that if one knows all the bits of d and $ed \leq N^2$, then one can factor in deterministic polynomial time.

Moreover, the result on RSA with small prime difference from de Weger [11] and one of the results for unbalanced RSA with small CRT-exponent by May [8] are also special cases of our method, as is an interesting situation not analyzed in the literature before, namely when both some MSBs and some LSBs of the private exponent are known. We will comment on these other applications of our general method in Appendix C.

5 Experiments

We state some experimental results to give an idea of the performance of our methods. In all the cases, $N \approx 2^{1024}$. The experiments are performed on a server containing two Pentium III processors of 1000 Mhz, and all the lattice basis reductions are done using Shoup's NTL [10].

For our MSB1 attack on small d, a typical case is $\beta = 0.3$, $\delta = 0.21$ (e.g. 70% of d is unknown). An attack using $m = 2$, $t = 1$ involved a 10 minute reduction of the 30-dimensional lattice.

For the MSB2 attack on small d, a typical case is $\beta = 0.6$, $\delta = 0.13$ (e.g. 22% of d is unknown). The attack using $m = 2$, $t = 2$ has a 50-dimensional lattice, that took $3\frac{1}{4}$ hours to reduce.

We performed the MSB2 attack on small e for $\alpha = 0.7$, $\delta = 0.08$ (e.g. 8% of d is unknown), using $m = 2$, $t = 2$. The reduction of the 50-dimensional lattice took $2\frac{3}{4}$ hours.

All typical cases are examples of our attacks where the bound on δ that we obtain in practice (for the low value $m = 2$) already exceeds the asymptotic bounds of other known attacks. More experimental results are included in Appendix D, where we also comment on how Assumption 1 holds in practice.

Last of all, we want to note that one could also apply the original method of Coppersmith described in [4] instead of Coron's reformulation [5]. In that case, the formulation of the method is a bit more technical, and the method produces essentially the same asymptotic bounds, but it has the advantage that the dimension of the lattice to reduce drops from a cubic to a quadratic function in m, which could significantly reduce the time necessary for LLL reduction.

References

1. Dan Boneh and Glenn Durfee: Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$, IEEE Transactions on Information Theory **46** [2000], 1339–1349.
2. Dan Boneh, Glenn Durfee and Yair Frankel: An Attack on RSA given a Small Fraction of the Private Key Bits, Proceedings of ASIACRYPT 1998, LNCS **1514** [1998], 25–34.
3. Johannes Blömer and Alexander May: New Partial Key Exposure Attacks on RSA, Proceedings of CRYPTO 2003, LNCS **2729** [2003], 27–43.
4. Don Coppersmith: Small Solutions to Polynomial Equations and Low Exponent RSA Vulnerabilities, Journal of Cryptology **10** [1997], 233–260.
5. Jean-Sébastien Coron: Finding Small Roots of Bivariate Integer Equations Revisited, Proceedings of EUROCRYPT 2004, LNCS **3027** [2004], 492–505.
6. Nick Howgrave-Graham: Finding Small Roots of Univariate Modular Equations Revisited, Cryptography and Coding, LNCS **1355** [1997], 131–142.
7. Arjen Lenstra, Hendrik Lenstra, Jr., and László Lovász: Factoring Polynomials with Rational Coefficients, Mathematische Ann. **261** [1982], 513–534.
8. Alexander May: Cryptanalysis of Unbalanced RSA with Small CRT-Exponent, Proceedings of CRYPTO 2002, LNCS **2442** [2002], 242–256.
9. Alexander May: Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring, Proceedings of CRYPTO 2004, LNCS **3152** [2004], 213–219.
10. Victor Shoup: NTL: A Library for doing Number Theory, online available at http://shoup.net/ntl
11. Benne de Weger: Cryptanalysis of RSA with Small Prime Difference, Applicable Algebra in Engineering, Communication and Computing **13** [2002], 17–28.
12. Michael Wiener: Cryptanalysis of Short RSA Secret Exponents, IEEE Transactions on Information Theory **36** [1990], 553–558.

A The Bound $X^{1+3\tau}Y^{2+3\tau}Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau}$

In this appendix, we show how to obtain the bound $X^{1+3\tau}Y^{2+3\tau}Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau}$ for the attack using f_{MSB1} (Section 4.1.1).

In Section 4.1.1, we described how to construct the lattice. The matrix containing the basis vectors is triangular and has the following diagonal entries (corresponding to the polynomials g, h, g' and h', respectively):

$X^mY^mZ^{m+t}$	for $i = 0, \ldots, m; j = 0, \ldots, m-i; k = 0, \ldots, j$
$X^mY^mZ^{m+t}$	for $i = 0, \ldots, m; j = 0, \ldots, m-i; k = j+1, \ldots, j+t$
$X^{m+i}Y^{m+j}Z^{m+t+k}W$	for $i = 0, \ldots, m+1; j = m+1-i; k = 0, \ldots, j$
$X^{m+i}Y^{m+j}Z^{m+t+k}W$	for $i = 0, \ldots, m+1; j = m+1-i; k = j+1, \ldots, j+t$

Since we have to optimize t in terms of m, we put $t = \tau m$. Elementary computations show that the dimension of L is

$$\omega = \frac{1}{6}(m^3(1+3\tau) + m^2(9+15\tau)) + o(m^2),$$

and that

$$\det(L) = X^{\frac{1}{6}(m^4(1+3\tau)+m^3(10+18\tau)+o(m^3))} \cdot Y^{\frac{1}{6}(m^4(1+3\tau)+m^3(11+18\tau)+o(m^3))} \cdot Z^{\frac{1}{6}(m^4(1+4\tau+3\tau^2)+m^3(10+27\tau+18\tau^2)+o(m^3))} \cdot W^{\frac{1}{6}(m^2(3+6\tau)+o(m^2))}.$$

When we apply LLL-reduction to our lattice, the polynomials $f_1(x, y, z)$ and $f_2(x, y, z)$ corresponding to the shortest two vectors in the reduced basis satisfy $f_1(x_0, y_0, z_0) \equiv 0 \bmod n$ and $f_2(x_0, y_0, z_0) \equiv 0 \bmod n$. In order to apply Howgrave-Graham's lemma, we explained in Section 3 that

$$\det(L) \leq (2^{\frac{-\omega}{4}} \frac{1}{\sqrt{\omega}})^{\omega-1} n^{\omega-1}.$$

must hold.

Ignoring terms that do not depend on $n = X^m Y^m Z^{m+t} W$, and ignoring terms of order $o(m^3)$ (we let these terms contribute to ϵ), we obtain that if

$$X^{m^4(1+3\tau)+m^3(10+18\tau)} Y^{m^4(1+3\tau)+m^3(11+18\tau)} Z^{m^4(1+4\tau+3\tau^2)+m^3(10+27\tau+18\tau^2)} \leq (XYZ^{1+\tau})^{m^4(1+3\tau)+m^3(9+15\tau)} W^{m^3(1+3\tau)}$$

the polynomials $f_1(x, y, z)$ and $f_2(x, y, z)$ satisfy the Howgrave-Graham bound. The condition above simplifies into

$$X^{1+3\tau} Y^{2+3\tau} Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau}.$$

B The Bound $X^{2+3\tau} Y^{3+6\tau+3\tau^2} Z^{3+3\tau} \leq W^{2+3\tau}$

In this appendix, we show how to obtain the bound $X^{2+3\tau} Y^{3+6\tau+3\tau^2} Z^{3+3\tau} \leq W^{2+3\tau}$ for the attack using f_{MSB2} (Section 4.1.2).

In Section 4.1.2, we described how to construct the lattice. The matrix containing the basis vectors is triangular and has the following diagonal entries (corresponding to g, h, g', and h'):

$X^m Y^{m+t} Z^m$	for $i = 0, \ldots, m; j = 0, \ldots, m-i; k = 0, \ldots, m-i$
$X^m Y^{m+t} Z^m$	for $i = 0, \ldots, m; j = m-i+1, \ldots, m-i+t;$ $k = 0, \ldots, m-i$
$X^{m+i} Y^{m+t+j} Z^{m+k} W$	for $i = 0, \ldots, m+1; j = 0, \ldots, m+t+1-i;$ $k = m+1-i$
$X^{m+i} Y^{m+t+j} Z^{m+k} W$	for $i = 0, \ldots, m; j = m+t+1-i; k = 0, \ldots, m-i$

One can check that

$$\dim(L) = \omega = \frac{1}{6}(m^3(2+3\tau) + m^2(15+15\tau)) + o(m^2),$$

and the determinant of L is equal to

$$X^{\frac{1}{6}(m^4(2+3\tau)+m^3(17+18\tau)+o(m^3))} \cdot Y^{\frac{1}{6}(m^4(2+5\tau+3\tau^2)+m^3(18+36\tau+18\tau^2)+o(m^3))} \cdot Z^{\frac{1}{6}(m^4(2+3\tau)+m^3(18+18\tau)+o(m^3))} \cdot W^{\frac{1}{6}(m^2(6+6\tau)+o(m^2))}.$$

When we apply LLL-reduction to our lattice, the polynomials $f_1(x, y, z)$ and $f_2(x, y, z)$ corresponding to the shortest two vectors in the reduced basis satisfy $f_1(x_0, y_0, z_0) \equiv 0 \bmod n$ and $f_2(x_0, y_0, z_0) \equiv 0 \bmod n$. In order to apply Howgrave-Graham's Lemma, it must hold that

$$\det(L) \leq (2^{\frac{-\omega}{4}} \frac{1}{\sqrt{\omega}})^{\omega-1} n^{\omega-1}.$$

Ignoring terms that do not depend on $n = X^m Y^{m+t} Z^m W$, and ignoring terms of order $o(m^3)$ (we let these terms contribute to ϵ), we obtain that if

$$X^{m^4(2+3\tau)+m^3(17+18\tau)} Y^{m^4(2+5\tau+3\tau^2)+m^3(18+36\tau+18\tau^2)} Z^{m^4(2+3\tau)+m^3(18+18\tau)} \leq (XY^{1+\tau}Z)^{m^4(2+3\tau)+m^3(15+15\tau)} W^{m^3(2+3\tau)}$$

the polynomials $f_1(x, y, z)$ and $f_2(x, y, z)$ satisfy Howgrave-Graham's bound. The condition above simplifies into

$$X^{2+3\tau} Y^{3+6\tau+3\tau^2} Z^{3+3\tau} \leq W^{2+3\tau}.$$

C Other Special Cases of Our Method

In this appendix, we show that results from [11] and [8] are also special cases of our method, as is the case where both MSBs and LSBs of d are known.

In the case of RSA with small prime difference, described by de Weger in [11], we have $p - q \leq N^\beta$, $k \leq d \leq N^\delta$ and $p + q - 2\sqrt{N} \leq N^{2\beta - \frac{1}{2}}$. The function $f_{dW}(x, y, z) = ex - y(N - 2\sqrt{N} - z) - 1$ has the same monomials as f_{MSB1}. When we substitute $X, Y = N^\delta$, $Z = N^{2\beta-\frac{1}{2}}$, we find that for $\beta < \frac{3}{4}$, we have $W = N^{1+\delta}$. Using $X^{1+3\tau} Y^{2+3\tau} Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau}$, we get $\delta \leq \frac{1}{6}(4\beta + 5) - \frac{1}{3}\sqrt{(4\beta+5)(4\beta-1)}$. To obtain de Weger's second bound, sublattices are needed.

Also, a bound for unbalanced RSA with small CRT-exponent by May [8] can be derived from our inequality belonging to f_{MSB1}. The setting is $ed_p - k(p - 1) - 1 = 0$, where $d_p \leq N^\delta$, $p \geq N^{1-\beta}$, and $k \leq N^{\beta+\delta}$. Multiplication with q yields $ed_pq - (k - 1)(N - q) - N = 0$. This gives us the polynomial $f_{Ma1} = ex - y(N - z) - N$. The upper bounds for the root $(x_0, y_0, z_0) = (d_pq, k - 1, q)$ are $X, Y = N^{\beta+\delta}$ and $Z = N^\beta$. Additionally, we have $W = N^{1+\beta+\delta}$. Plugging these values in our inequality, we find the bound $\delta \leq 1 - \frac{2}{3}(\beta + \sqrt{3\beta + \beta^2})$.

The last special case we describe in this appendix is the the situation where both MSBs and LSBs of an exponent d are known. Let d_L be a known LSB part of size N^κ of the key d, followed by an unknown middle part x of size N^δ, which itself is followed by a known MSB part d_M, of size $N^{\beta-\kappa-\delta}$. Hence, we can write d as $d = d_L + M_1(x + M_2 d_M)$, where $M_1 \geq N^\kappa$, and $M_2 \leq N^\delta$. Note that $\kappa = 0$ describes the case where only MSBs are known, whereas $\kappa = \beta - \delta$ corresponds to the LSB scenario.

When we omit partial knowledge of k, the function $f_{MSB+LSB1}(x,y,z) = eM_1x - Ny + yz + R$, with $R = ed_L + eM_1M_2d_M - 1$, has the small root $(x_0, y_0, z_0) = (x, k, p+q-1)$, with $X = N^{\delta}$, $Y = N^{\beta}$, and $Z = 3N^{\frac{1}{2}}$.

As the function has the same monomials as f_{MSB1}, one can use the same inequality to conclude that the attack works for $\delta < \frac{5}{6} - \frac{1}{3}\sqrt{1+6\beta}$. Hence, the result is exactly the same as when only MSBs or only LSBs are known and knowledge of k is not used. Apparently, as long as the unknown part of d is connected, its place does not make a difference, only its length.

When we use the partial knowledge of k provided by the approximation $\tilde{k}$, we obtain the function $f_{MSB+LSB2}(x,y,z) = eM_1x - Ny + yz + \tilde{k}z + R$, with $R = ed_L + eM_1M_2d_M - \tilde{k}N - 1$.

Analysis similar to the f_{MSB2} case shows that if $\gamma = \max\{\delta+\kappa, \beta - \frac{1}{2}\} = \delta + \kappa$, we obtain the bound $\delta \leq \frac{3-4\kappa-4\kappa^2}{16+16\kappa}$, valid for $\beta \leq \frac{11+4\kappa-4\kappa^2}{16+16\kappa}$. In the case $\gamma = \beta - \frac{1}{2}$, we find that the attack works whenever $\delta \leq \frac{1}{3} + \frac{1}{3}\beta - \frac{1}{3}\sqrt{4\beta^2+2\beta-2}$, valid for $\beta \geq \frac{11+4\kappa-4\kappa^2}{16+16\kappa}$.

Naturally, equivalent bounds can be derived when d is full size and e is not.

D More Experimental Results

In addition to Section 5, we now show more experimental results. The experiments we did for this appendix are only for $m = 1$ and $m = 2$, which means the lattices are relatively small and the lattice reduction can be performed in a matter of seconds or minutes. In the full version of this paper, experiments for larger parameters will be included.

As in Section 5, the experiments are performed on a server containing two Pentium III processors of 1000 Mhz, and all the lattice basis reductions are done using Shoup's NTL [10]. In contrast to the experimental results mentioned in Section 5, we assume here that we have a 256 bit modulus. So one has to keep in mind that for $N \approx 2^{1024}$, the running time of the LLL-procedure will be longer.

As the bounds on δ stated in Theorem 1 and 2 are asymptotic bounds, the goal of the tables in this appendix is to provide some intuition of what bounds on δ our attacks can achieve in practice. For example, the table in Fig. 5 shows that for $\beta = 0.3$, the asymptotic bound of the attack using f_{MSB1} from Section 4.1.1 is $\delta < 0.28 - \epsilon$. When we use the parameters $m = 2$, $t = 1$, our attack works for $\delta < 0.21$. The attack involves a lattice of dimension 30, which takes approximately 25 seconds to reduce.

This example is one of the three so-called 'typical cases' of Section 5. These are examples where the bound on δ that we obtain in practice exceeds the asymptotic bounds of other known attacks. In the tables in this appendix, the typical cases are written in bold.

For the choice of t, recall from Section 4.1.1 that $t = \tau m$, and that we use $\tau = \frac{1}{2} - \delta$ to obtain the asymptotic result of our attack using f_{MSB1}. This explains that for $m = 1$ in Fig. 5, a value of t larger than 1 gives no significant improvement, but for $m = 2$, $t = 2$ may give a better result when the bound on

β	δ	$m=1$			$m=2$		
	asympt.	$t=0$	$t=1$	$t=2$	$t=0$	$t=1$	$t=2$
0.30	**0.28**	0.19	0.19	0.19	0.19	**0.21**	0.21
0.35	0.25	0.13	0.14	0.14	0.14	0.16	0.16
0.40	0.22	0.09	0.11	0.11	0.09	0.14	0.15
0.45	0.19	0.04	0.10	0.10	0.05	0.12	0.12
0.50	0.17	0	0.08	0.09	0	0.10	0.11
0.55	0.14	0	0.08	0.08	0	0.09	0.11
0.60	0.12	0	0.04	0.04	0	0.06	0.10
0.65	0.10	0	0	0	0	0	0.06
0.70	0.07	0	0	0	0	0	0.01
0.75	0.05	0	0	0	0	0	0
0.80	0.03	0	0	0	0	0	0
0.85	0.01	0	0	0	0	0	0
Dimension:		10	16	22	20	**30**	40
LLL (sec):		1	2	8	3	**25**	100

Fig. 5. Experiments f_{MSB1} for small d

β	δ	$m=1$				$m=2$		
	asympt.	$t=0$	$t=1$	$t=2$	$t=3$	$t=0$	$t=1$	$t=2$
0.30	0.19	0.19	0.20	0.20	0.20	0.19	0.19	0.19
0.35	0.19	0.15	0.16	0.16	0.16	0.16	0.16	0.16
0.40	0.19	0.12	0.12	0.12	0.12	0.14	0.15	0.15
0.45	0.19	0.10	0.11	0.12	0.12	0.12	0.13	0.13
0.50	0.19	0.08	0.11	0.12	0.12	0.12	0.13	0.13
0.55	0.19	0.08	0.11	0.12	0.12	0.11	0.13	0.13
0.60	**0.19**	0.05	0.11	0.11	0.11	0.11	0.12	**0.13**
0.65	0.19	0	0.05	0.06	0.06	0.05	0.08	0.10
0.70	0.18	0	0	0	0	0	0.04	0.05
0.75	0.14	0	0	0	0	0	0	0
0.80	0.11	0	0	0	0	0	0	0
0.85	0.08	0	0	0	0	0	0	0
0.90	0.05	0	0	0	0	0	0	0
0.95	0.03	0	0	0	0	0	0	0
Dimension:		14	20	26	32	30	40	**50**
LLL (sec):		1	7	17	40	26	180	**480**

Fig. 6. Experiments f_{MSB2} for small d

α	δ	$m=1$				$m=2$		
	asymptotic	$t=0$	$t=1$	$t=2$	$t=3$	$t=0$	$t=1$	$t=2$
0.50	0.50	0.25	0.33	0.38	0.40	0.32	0.37	0.41
0.55	0.33	0.17	0.21	0.23	0.25	0.21	0.23	0.24
0.60	0.27	0.09	0.14	0.17	0.18	0.13	0.16	0.19
0.65	0.22	0.02	0.07	0.10	0.10	0.07	0.11	0.13
0.70	0.18	0	0.02	0.03	0.04	0.02	0.04	**0.08**
0.75	0.14	0	0	0	0	0	0.01	0.02
0.80	0.11	0	0	0	0	0	0	0
0.85	0.08	0	0	0	0	0	0	0
0.90	0.05	0	0	0	0	0	0	0
0.95	0.03	0	0	0	0	0	0	0
Dimension:		14	20	26	32	30	40	**50**
LLL (sec):		1	5	13	40	33	180	**520**

Fig. 7. Experiments f_{MSB2} for small e

δ is 'low'. For the attacks using f_{MSB2} (Section 4.1.2 and 4.2), $\tau = \frac{\frac{1}{2}-\delta-\gamma}{2\gamma}$. This explains for example, that when $e = N^\alpha$ with α close to $\frac{1}{2}$, using a larger t gives a better bound on δ in the experiments (as can be seen in Fig. 7).

Having done some experiments, we can now comment on Assumption 1. Let $g(x, y, z)$ and $h(x, y, z)$ be polynomials that correspond to LLL-reduced vectors in our method, for which Howgrave-Graham's bound is satisfied. If $g(x_0, y_0, z_0) = h(x_0, y_0, z_0) = 0$, but the resultant computations with g and h yield the zero-polynomial, then Assumption 1 does not hold. Therefore, we performed some tests to see how often this occurs. We found that for approximately 0.1% of pairs (g, h) the heuristic failed. This does not mean that the method will always fail in these cases. Usually, there are several vectors that satisfy Howgrave-Graham's bound, hence if one pair fails, other pairs can yield the solution.

Experiments also show that the theoretical bound under which our methods works, $\det(L) \leq (2^{\frac{-\omega}{4}} \frac{1}{\sqrt{\omega}})^{\omega-1} n^{\omega-1}$, is far too strict. It would imply that for $m \in \{1, 2\}$, the method will never work, which clearly contradicts the practice. This is both due to the term $(2^{\frac{-\omega}{4}} \frac{1}{\sqrt{\omega}})^{\omega-1}$, when it is known that LLL-reduction achieves much better bounds in practice, and to the fact that we use the LLL-bound for the second smallest reduced vector. In practice, we experienced that our method works until we come close to $\det(L) \leq n^\omega$ (the bound for the first reduced vector to be small enough, omitting the constant term).

The RSA Group is Pseudo-Free

Daniele Micciancio

Department of Computer Science and Engineering,
University of California at San Diego, La Jolla CA 92093, USA
daniele@cs.ucsd.edu
http://www.cse.ucsd.edu/users/daniele

Abstract. We prove, under the strong RSA assumption, that the group of invertible integers modulo the product of two safe primes is pseudo-free. More specifically, no polynomial time algorithm can output (with non negligible probability) an unsatisfiable system of equations over the free abelian group generated by the symbols $g_1, \ldots, g_n$, together with a solution modulo the product of two randomly chosen safe primes when $g_1, ..., g_n$ are instantiated to randomly chosen quadratic residues. Ours is the first provably secure construction of pseudo-free abelian groups under a standard cryptographic assumption, and resolves a conjecture of Rivest (TCC 2004).

1 Introduction

The notion of "pseudo-free group", put forward by Hohenberger in [10] and subsequently refined by Rivest [20], informally describes a finite computational group (i.e. a group that admits an efficient algorithmic implementation) with the security property that it is computationally hard to find solutions to any non-trivial equation over the group. More specifically, Rivest [20] defines pseudo-free (abelian) groups as computational (commutative) groups such that no polynomial time adversary, given random group elements $g_1, \ldots, g_n$ (chosen using to an appropriate sampling procedure), can output (with non negligible probability) an equation which is unsatisfiable over the free abelian group generated by the symbols $g_1, \ldots, g_n$, together with a solution to the equation in the computational group. As shown in [20], pseudo-freeness is a very strong assumption, and it implies many other computational assumptions typically used in cryptography, like the hardness of computing discrete logarithms and the RSA assumption in its standard and strong version. Each of these computational assumptions corresponds to a specific class of equations, e.g., the strong RSA assumption asserts that it is computationally infeasible to come up with an equation of the form $x^e = g$ (which is unsatisfiable over the free group $\{g^i : i \in \mathbb{Z}\}$ for $e > 1$) together with a solution $x = h$ such that $h^e = g$ in the multiplicative group $\mathbb{Z}_N^*$ of the invertible integers modulo the product $N = PQ$ of two large primes.

Free groups are widely used in computer science, and most modern cryptography relies on the hardness of computational problems over finite groups. So,

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 387–403, 2005.

as argued in [20], pseudo-free groups are a very interesting notion from a cryptographic perspective. For example, (non abelian) free groups are used in the so called Dolev-Yao model [7] for the symbolic analysis of public key cryptographic protocols. In the last few years, there have been several efforts to bridge the gap between the symbolic model of [7] (typically used in the area of formal methods for the analysis of security protocols) and the standard computational model used in cryptography (see for example [1, 18, 19, 17, 13, 11, 3]) with the goal of proving computational soundness results for symbolic analysis methods. An interesting question is whether pseudo-free groups can be used to extend (in a computationally sound way) the Dolev-Yao security model (in which encryption and decryption are viewed as black-box operations with no algebraic properties) with richer data structures and cryptographic functions (e.g., homomorphic encryption schemes) that make fundamental use of computational groups. Other motivations for studying pseudo-free groups mentioned in [20] are the following:

- Using a stronger assumption (that subsumes many other common cryptographic assumptions, like the hardness of computing discrete logarithms, and the strong RSA assumption) may make proofs easier.
- As the strong RSA assumption has been very useful in the construction of many cryptographic functions [8, 4, 6] which are not known to be secure under the standard version of the RSA assumption, assuming that a group is pseudo-free may allow an even wider range of applications.
- Pseudo-freeness has been linked [10] to the construction of specific cryptographic primitives, like directed transitive signature schemes, for which no solution is currently known.

The main question left open by Rivest in [20] is: do pseudo-free groups exist?

In [20] Rivest suggested the RSA group $\mathbb{Z}_N^*$ (where $N = PQ$ is the product of two large primes) as a possible candidate pseudo-free abelian group, and nicknamed the corresponding conjecture the "super-strong RSA assumption". In this paper we resolve Rivest's conjecture and prove that $\mathbb{Z}_N^*$ is pseudo-free under the strong RSA assumption, at least when $N = PQ$ is the product of two "safe primes" (i.e., odd primes such that $p = (P-1)/2$ and $q = (Q-1)/2$ are also prime[1]), a special class of prime numbers widely used in cryptography. In other words, we prove that if the strong RSA assumption holds true, then the super-strong RSA assumption also holds. Our result is the first example of provably secure pseudo-free group based on a standard cryptographic assumption. In fact, we prove that the RSA group satisfies an even stronger version of the pseudo-freeness property than the one defined in [20]: we show that no adversary can efficiently compute an unsatisfiable *system* of equations (as opposed to a single equation) together with a solution in the given computational group.

Our proof is based on a rewriting process that, starting from an arbitrary equation (or system of equations), yields simpler and simpler equations with the following properties:

[1] Equivalently, using more standard mathematical terminology, $P = 2p + 1$ and $Q = 2q + 1$ where p and q are *Sophie-Germain* primes.

– unsatisfiable equations over the free group are mapped to unsatisfiable equations over the free group, and
– solutions to the original equations (over a computational group) can be efficiently mapped to solutions to the resulting equations (over the same computational group).

Some of our transformations work for arbitrary groups and might be of independent interest. For example, we show how to transform systems of equations into a single equation (Theorem 4), how to map equations in several variables to univariate equations (Lemma 3), and how to map unsatisfiable equations of the form $x^e = g^d$ (where e and d are arbitrary integers) to equations where the exponents are of the form $e = c^{h+1}$ and $d = c^h$ (Theorem 3).

Organization. The rest of the paper is organized as follows. In Section 2 we introduce basic definitions and notation for equations and groups. In Section 3 we prove that the RSA group satisfies the basic definition of pseudo-free group (involving a single equation). In Section 4 we extend the result to systems of equations. Section 5 concludes with a discussion of open problems.

2 Preliminaries

A group is an algebra with a binary associative operation $\circ$, a unary operation $()^{-1}$ (inverse) and a constant 1 (identity) satisfying the equational axioms $(x \circ y) \circ z = x \circ (y \circ z)$, $x \circ 1 = 1 \circ x = x$, and $x \circ (x)^{-1} = (x)^{-1} \circ x = 1$. A group is abelian if the operation $\circ$ is commutative, i.e., it also satisfies $x \circ y = y \circ x$. In this paper we are interested in computational groups, i.e., groups that admit an efficient algorithmic implementation.

Definition 1. *A computational group (associated to group $(G, \circ, ()^{-1}, 1)$) is defined by a mapping $\langle\cdot\rangle : G \to \{0,1\}^*$ (the representation function) such that the following operations can be performed in polynomial time:*

– *Test membership in $\langle G\rangle$, i.e., given a string x, determine if it is a valid representation of a group element.*
– *Given $\langle x\rangle$ and $\langle y\rangle$, compute $\langle x \circ y\rangle$.*
– *Given $\langle x\rangle$, compute $\langle x^{-1}\rangle$.*
– *Compute the representation of the group identity element $\langle 1\rangle$.*
– *Sample a group element $\langle g\rangle \in \langle G\rangle$ (with not necessarily uniform probability distribution.)*

For simplicity, in the definition above, we focused on computational groups in which each group element has a unique representation, as all the computational groups studied in this paper have this property. (The definition of computational group can be easily extended to cases where group elements may have multiple representations, by introducing an efficiently computable equivalence relation on group representations.) The requirement that membership in $\langle G\rangle$ be efficiently decidable is also not strictly necessary, but convenient, and all computational

groups studied in this paper have this property. Also, sometimes the definition of computational group requires the distribution output by the sampling algorithm $\langle g \rangle \in \langle G \rangle$ to be uniform over G, while other times no sampling algorithm is required at all. In this paper $\langle g \rangle \in \langle G \rangle$ is an arbitrary sampling procedure, which is used to generate nontrivial group elements.

For brevity, we identify computational groups with the underlying mathematical group, and write $x \circ y$, x^{-1}, etc., to denote the corresponding operation on the representations of the group elements. Also, we use multiplicative notation xy for the group binary operation $x \circ y$, and use exponential notation x^n to denote the n-fold composition of x with itself. Formally, x^n is defined inductively by the rules $x^0 = 1$, $x^{n+1} = x \circ x^n$. The notation is extended to negative exponents in the obvious way $x^{-n} = (x^n)^{-1}$.

For any set of symbols A, the free abelian group $\mathcal{F}(A)$ generated by A is the initial algebra with constant symbols A satisfying the abelian group equations. It is easy to see that each group element has a unique representation as a product $\prod_{a \in A} a^{d_a}$, where $d_a \in \mathbb{Z}$ for all $a \in A$.

Let X and A be two disjoint finite sets of variable and constant symbols. We define $X^{-1} = \{x^{-1} : x \in X\}$ and $A^{-1} = \{a^{-1} : a \in A\}$. A group equation over variables X and constants A, is a pair $E = (w_1, w_2)$ of words (usually written as $E : w_1 = w_2$) over the alphabet $X \cup X^{-1} \cup A \cup A^{-1}$. Unless otherwise specified, we interpret E as an equation over the free group $\mathcal{F}(A)$. A solution to $E : w_1 = w_2$ (over the free group $\mathcal{F}(A)$) is a function $\sigma : X \to \mathcal{F}(A)$ such that $\sigma(w_1) = \sigma(w_2)$, where σ is extended to words over $X \cup X^{-1} \cup A \cup A^{-1}$ homomorphically in the obvious way. We say that an equation $E : w_1 = w_2$ is satisfiable (over the free group) if it admits a solution. We say it is unsatisfiable otherwise.

Let G be a (computational) group. A group equation over G (denoted E_α) is defined by an equation E over variables X and constants A, and a function $\alpha : A \to G$. A solution to equation $E_\alpha : w_1 = w_2$ is a function $\xi : X \to G$ such that $(\alpha \cup \xi)(w_1) = (\alpha \cup \xi)(w_2)$.

From a computational point of view, we assume that equations $E : w_1 = w_2$ are represented using compact notation for exponential expressions a^i, so that exponentially large exponents i can be stored in polynomial space. This is easily seen to be equivalent to many other formalisms to compactly represent terms w_1, w_2, like, for example, the straight-line programs used in [10].

2.1 Statistical Distance

A function f is negligible if it decreases faster than any inverse polynomial, i.e., for any $c > 0$ there is an n_0 such that $|f(n)| \leq 1/n^c$ for any $n > n_0$.

Let X and Y be two discrete random variables over a (countable) set A. The statistical distance between X and Y is the quantity

$$\Delta(X, Y) = \frac{1}{2} \sum_{a \in A} |\Pr\{X = a\} - \Pr\{Y = a\}|.$$

In this paper we use the fact that for any two random variables X and Y over set A, and predicate $p : A \to \{0, 1\}$,

$$|\Pr[p(X) = 1] - \Pr[p(Y) = 1]| \leq \Delta(X, Y).$$

In particular, if $p(X)$ happens with non negligible probability, and $\Delta(X, Y)$ is negligible, then also $p(Y)$ happens with non negligible probability.

2.2 Pseudo-Free Groups

Intuitively, a computational group is pseudo-free if no efficient algorithm can find a nontrivial relation among randomly chosen group elements, i.e., an equation (or system of equations) which is unsatisfiable over the free group, together with a solution over the computational group. Since for any finite group G, the equation $x^{|G|+1} = a$ is unsatisfiable over the free group $\mathcal{F}(\{a\})$, but has solution $x = a$ over G, in order to properly define pseudo-free groups we need to consider families of groups $\{G_N\}$ where N is chosen at random from a probability ensemble $\mathcal{N}$. In particular, given a randomly chosen $N \in \mathcal{N}$, the order of the group $o(G_N)$ should be hard to compute.

Definition 2. *A family of computational groups $\mathcal{G} = \{G_N\}_{N \in \mathcal{N}}$ is pseudo-free if for any set A of polynomial size $|A| = p(k)$ (where k is a security parameter), and probabilistic polynomial (in k) time algorithm $\mathcal{A}$, the following holds. Let $N \in \mathcal{N}(k)$ be a randomly chosen group index, and $\alpha: A \to G_N$ a function defining $|A|$ group elements chosen independently at random according to the computational group sampling procedure. Then, the probability that $\mathcal{A}(N, \alpha) = (E, \xi)$ outputs an unsatisfiable equation E (over variables X and constants A) together with a solution $\xi: X \to G_N$ to E_α over G_N, is negligible.*

2.3 The RSA Group

In this paper, we study the group $\mathbb{Z}_N^*$ of invertible integers modulo N. This is a computational group, with the usual representation of each group element as an integer in $\{0, \ldots, N-1\}$. Membership $g \in \mathbb{Z}_N^*$ can be easily tested by computing $\gcd(g, N)$ and checking that $\gcd(g, N) = 1$. The group $\mathbb{Z}_N^*$ can be efficiently sampled uniformly at random by picking an integer $g \in \{0, \ldots, N-1\}$ with uniform distribution, and checking if $g \in \mathbb{Z}_N^*$. However, in this paper, it is more convenient to consider the computational group $\mathbb{Z}_N^*$ together with a different sampling procedure that chooses g at random from a subgroup of $\mathbb{Z}_N^*$. An element $g \in \mathbb{Z}_N^*$ is called a quadratic residue if $g = h^2 \bmod N$ for some $h \in \mathbb{Z}_N^*$. The set of quadratic residues modulo N is denoted QR_N, and it is a subgroup of $\mathbb{Z}_N^*$. The subgroup QR_N can be efficiently sampled by picking $h \in \mathbb{Z}_N^*$ uniformly at random and setting $g = h^2 \bmod N$. Unless otherwise specified, in this paper we always consider the computational group $\mathbb{Z}_N^*$ with this sampling procedure that selects g uniformly at random from QR_N.

When $N = P \cdot Q$ is the product of two prime numbers, $\mathbb{Z}_N^*$ is commonly called an RSA group, after the encryption function of Rivest, Shamir and Adleman [21], which started a widespread use of these groups in cryptography. In this paper we are interested in RSA groups where P and Q are primes of special form. A prime number p is called a Sophie-Germain prime if $2p + 1$ is also prime. In the

cryptographic literature, the number $2p+1$ (where p is a Sophie-Germain prime) is usually called a safe prime. In other words, a safe prime $P = 2p + 1$ is an odd prime number such that $p = (P - 1)/2$ is also prime. Safe primes are relatively easy to find in practice (e.g., by choosing p at random and testing p and $2p+1$ for primality), although there is no known mathematical proof showing that there are infinitely many of them. Safe primes are widely used in cryptography. For example, the RSA group $\mathbb{Z}_N^*$ where $N = P \cdot Q$ is the product of two safe primes has been used in [8, 9, 6].

Let $\mathcal{N}$ be the set of all safe prime products. We assume some standard probability distribution on $\mathcal{N}$, as typically used in many cryptographic applications. (E.g., choose N as the product of two random k-bit safe primes.) The following computational problems are considered hard, and have been used as the basis for many cryptographic applications:

- Factoring problem: given an integer $N \in \mathcal{N}$, compute prime factors P, Q such that $N = P \cdot Q$;
- RSA problem: given an integer $N \in \mathcal{N}$, an integer e relatively prime with $\phi(N) = (P - 1)(Q - 1)$, and a randomly chosen group element $\gamma \in \mathbb{Z}_N^*$, compute a $\xi \in \mathbb{Z}_N^*$ such that $\xi^e = \gamma \bmod N$;
- Strong RSA problem: given an integer $N \in \mathcal{N}$, and a randomly chosen group element $\gamma \in \mathbb{Z}_N^*$, output an integer $e > 1$ and a group element $\xi \in \mathbb{Z}_N^*$ such that $\xi^e = \gamma \bmod N$.

In this paper we are primarily interested in the strong RSA problem and its relation to pseudo-freeness. It is convenient to consider the following variant of the strong RSA problem where the input γ is chosen as a random quadratic residue:

- Strong QR-RSA problem: given an integer $N \in \mathcal{N}$, and a randomly chosen quadratic residue $\gamma \in QR_N$, output an integer $e > 1$ and a group element $\xi \in \mathbb{Z}_N^*$ such that $\xi^e = \gamma \bmod N$.

It can be easily shown [6] that this variant is not any easier than the standard strong RSA problem.

Theorem 1 (See [6], Section 4). *If the strong RSA problem modulo a safe prime product is hard, then the strong QR-RSA problem modulo a safe prime product is also hard.*

For any prime product $N = P \cdot Q$, the group $\mathbb{Z}_N^*$ has cardinality $o(\mathbb{Z}_N^*) = \phi(N) = (P - 1)(Q - 1)$ and it is isomorphic to $\mathbb{Z}_P^* \times \mathbb{Z}_Q^*$, with isomorphism given by $\xi \mapsto (\xi \bmod P, \xi \bmod Q)$. If $P = 2p + 1$ and $Q = 2q + 1$ are safe primes, the group $\mathbb{Z}_N^*$ has order $4pq$, and the subgroup $QR_N \subset \mathbb{Z}_N^*$ has order $o(QR_N) = pq$. In particular, all elements in QR_N have order[2] $1, p, q$ or pq.

[2] The order of an element γ in a group G is the smallest positive integer $o(\gamma) \geq 1$ such that $\gamma^{o(\gamma)} = 1$.

3 The RSA Group is Pseudo-Free

In this section we prove, under the strong RSA assumption, that the RSA group $\mathbb{Z}_N^*$ (where N is the product of two safe primes, and elements are sampled uniformly at random from QR_N) is pseudo-free.

Theorem 2. *Let $\mathcal{N}$ be a distribution over safe prime products such that the strong RSA problem modulo $N \in \mathcal{N}$ is hard. Then the family of computational groups $\mathbb{Z}_N^*$ of invertible integers modulo $N \in \mathcal{N}$ (with the modular multiplication group operation, and uniform sampling procedure over QR_N) is pseudo-free.*

Proof. Assume that $\mathbb{Z}_N^*$ is not pseudo-free, i.e., there is a probabilistic polynomial time algorithm $\mathcal{A}$ that on input a randomly chosen $N \in \mathcal{N}(k)$ and random group elements $\alpha: A \to QR_N$ (for some polynomial sized set A), outputs an unsatisfiable equation $E: w_1 = w_2$ (over constants A and variables X) together with a solution $\xi: X \to \mathbb{Z}_N^*$ to E_α over the group $\mathbb{Z}_N^*$. We use $\mathcal{A}$ to solve the strong QR-RSA problem for the same distribution of the modulus N. Namely, given a randomly chosen $N \in \mathcal{N}(k)$ and $\gamma \in QR_N$, we compute an integer $e > 1$ and group element $\xi \in \mathbb{Z}_N^*$ such that $\xi^e = \gamma$. By Theorem 1 this also implies an algorithm to solve the standard strong RSA problem.

The reduction works as follows. Let (N, γ) be an instance of the strong QR-RSA problem. We begin by checking if γ is a generator for QR_N. This can be easily done using the following lemma.[3]

Lemma 1. *Let $N = P \cdot Q$ be the product of two distinct safe primes, and $\gamma \in QR_N$ a quadratic residue. Then γ is a generator for QR_N if and only if* $\gcd(\gamma - 1, N) = 1$.

If γ is not a generator for QR_N, then we can easily solve the strong QR-RSA problem instance (N, γ) as described below. Given N and $\gamma \in QR_N$, we compute $g = \gcd(\gamma - 1, N)$. Since $N = PQ$, it must be $g \in \{1, P, Q, PQ\}$. We distinguish three cases.

- If $g = PQ = N$, then N divides $\gamma - 1$, and $\gamma = 1 \pmod N$. So, we can immediately output a solution to the strong QR-RSA input problem (N, γ), e.g., $(\xi, e) = (1, 3)$.
- If $g \in \{P, Q\}$, then we can easily compute $\phi(N) = (P-1) \cdot (Q-1) = (g-1)(N/g-1)$. This also easily yields a solution $(\xi, e) = (\gamma, \phi(N) + 1)$ to the strong QR-RSA problem (N, γ).
- If $g = 1$, then by Lemma 1 γ is a generator for QR_N and we proceed as follows.

In the rest of the proof we assume that γ is a generator of QR_N. We use γ to sample the group elements $\alpha(a) \in QR_N$ and generate an input instance (N, α) for algorithm $\mathcal{A}$. Since $\mathcal{A}$ works only with non negligible probability, we

[3] The proof of this and other lemmas are omitted because of space limitations. All proofs can be found in the full version of the paper on the author's web page.

need the input values $\alpha(a)$ to be distributed (almost) uniformly at random over QR_N. The following lemma shows that γ can be used to sample QR_N almost uniformly at random.

Lemma 2. *For any cyclic group G and generator $\gamma \in G$, if $v \in \{0, \ldots, B-1\}$ is chosen uniformly at random, then the statistical distance between γ^v and the uniform distribution over G is at most $|G|/2B$.*

For any $a \in A$, choose $v_a \in [0, \ldots, N \cdot |A| \cdot K - 1]$ uniformly at random for some super-polynomial function $K(k) = k^{\omega(1)}$, and set $\alpha(a) = \gamma^{v_a}$. By Lemma 2, the statistical distance between $\alpha(a)$ and the uniform distribution over QR_N is at most $|QR_N|/2N|A|K \leq 1/|A| \cdot K$. Since the values $\alpha(a)$ are independently chosen, the statistical distance between α and a uniformly chosen assignment is at most $1/2K = k^{-\omega(1)}$.

Invoke algorithm $\mathcal{A}$ on input (N, α). We know that when α is distributed uniformly at random, algorithm $\mathcal{A}$ is successful with non negligible probability $\delta(k) = k^{-O(1)}$. Since α is within negligible statistical distance $1/K(k)$ from uniform, $\mathcal{A}$ succeeds on input α at least with non negligible probability $\delta(k) - 1/K(k)$. In the rest of the proof, we assume $\mathcal{A}$ is successful, and we consider the conditional success probability of the reduction. We will show that the conditional success probability is at least 1/3.

Fix the value of N, generator $\gamma \in QR_N$, and input (N, α) passed to algorithm $\mathcal{A}$. Let $E : w_1 = w_2$ and ξ be the equation and solution to E_α returned by $\mathcal{A}$. Remember that, for every $a \in A$, $\alpha(a) = \gamma^{v_a}$ for a randomly chosen $v_a \in \{0, \ldots, N \cdot |A| \cdot K - 1\}$. For any $a \in A$, let $w_a = v_a \bmod pq$ and $z_a = (v_a - w_a)/pq$. We remark that although the values v_a are known, and w_a, z_a are uniquely determined by v_a, the values w_a and z_a cannot be easily computed from v_a because the product pq is not known. Therefore, the values w_a and z_a cannot be used in the reduction process. We will use w_a and z_a only in the analysis of the reduction.

Notice that, given w_a, the conditional distribution of z_a is uniform over the set

$$S_a = \{0, \ldots, \lfloor (N|A|K - 1 - w_a)/pq \rfloor\}. \tag{1}$$

Also, given w_a, $\alpha(a) = \gamma^{v_a} = \gamma^{w_a}$ is uniquely determined, and z_a is uniformly distributed over the set S_a independently from α, E and ξ. In particular, the integers $z_a \in S_a$ are uniformly distributed independently from the success of algorithm $\mathcal{A}$.

Assume that $\mathcal{A}$ is successful, i.e., E is unsatisfiable over $\mathcal{F}(A)$, and $\xi : X \to QR_N$ is a valid solution to E_α. We use equation E and solution ξ to solve the original strong QR-RSA problem (N, γ). This is done in two steps. First, we transform equation E and solution ξ to E_α, into a new unsatisfiable equation E' and solution ξ' to E'_α containing only one variable symbol. Then, E' and ξ' are used to solve the strong QR-RSA problem (N, γ).

The equation and solution (E, ξ) is transformed into a univariate equation and solution (E', ξ') using the following lemma.

Lemma 3. *For any computational group G, there is a polynomial time algorithm that on input an equation E over constants A and variables X, and a variable assignment $\xi : X \to G$, outputs a univariate equation E' and value $\xi' \in G$, such that*

- *if E is unsatisfiable over the free group $\mathcal{F}(A)$, then E' is also unsatisfiable over $\mathcal{F}(A)$; and*
- *for any assignment $\alpha : A \to G$, if ξ is a solution to E_α then ξ' is a solution to E'_α.*

At this point we have an unsatisfiable equation of the form $E' : x^e = \prod_a a^{d_a}$ and a solution $\xi' \in \mathbb{Z}_N^*$ to E'_α. Notice that E' is satisfiable over the free group $\mathcal{F}(A)$ if and only if $e | \gcd(d_a : a \in A)$. So, it must be $e \not| \gcd(d_a : a \in A)$. Also, from the definition of $\alpha(a)$, we know that

$$(\xi')^e = \prod_a \alpha(a)^{d_a} = \gamma^{\sum_a v_a d_a}. \tag{2}$$

Assume without loss of generality that $e \geq 0$ and $d = \sum_a v_a d_a \geq 0$. (Otherwise, change the sign of e and/or the d_a for all $a \in A$, and possibly replace ξ' with $(\xi')^{-1}$ in order to satisfy (2).) In the rest of the proof we distinguish various cases, depending on the value of $\gcd(e, pq)$.

- If $\gcd(e, pq) = pq$ and $e \neq 0$, then we can immediately output the solution $(\gamma, e+1)$ to the strong QR-RSA problem (N, γ) because $o(\gamma) = pq$ and $\gamma^{e+1} = \gamma \cdot \gamma^e = \gamma \pmod N$. We remark that, although we cannot compute $\gcd(e, pq)$ (or even check if $\gcd(e, pq) = pq$) because pq is not know, we can guess that this is the case, and simply check if $\gamma, e+1)$ is indeed a solution to the given strong QR-RSA problem. Similar remarks apply to the other cases below.
- If $\gcd(e, pq) \in \{p, q\}$, then $o(\gamma^e) = pq/\gcd(e, pq) \in \{p, q\}$. In particular, γ^e is not a generator of QR_N, and, by Lemma 1, $\gcd(\gamma^e - 1, N) \neq 1$. Since $\gamma^e \neq 1 \pmod N$, we also have $\gcd(\gamma^e - 1, N) \neq N$. Therefore, it must be $g = \gcd(\gamma^e - 1, N) \in \{P, Q\}$. So, we can compute $\phi(N) = (P-1)(Q-1) = (g-1)(N/g-1)$, and output the solution $(\gamma, \phi(N)+1)$ to the strong QR-RSA problem (N, γ).
- The remaining cases are when $e = 0$ or $\gcd(e, pq) = 1$, and are described below.

If $e = 0$, Lemma 4 below shows that $d = 0$ with probability at most $1/2$. It follows that with probability at least $1/2$, $(\gamma, d+1)$ is a solution to the strong QR-RSA problem (N, γ) because $d + 1 > d \geq 1$ and $\gamma^{d+1} = \gamma \cdot \gamma^d = \gamma \cdot \xi^0 = \gamma$. So, the conditional success probability of the reduction is at least $1/2$.

Lemma 4. *The conditional probability (given α, $e = 0$, and $\{d_a : a \in A\}$ such that $e \not| \gcd\{d_a : a \in A\}$) that $d = \sum_a v_a d_a \neq 0$ is at least $1/2$.*

The last case to consider is when $\gcd(e, pq) = 1$. This is the most complicated of all cases. This time, we first show that $e|d$ with probability at most $2/3$.

Lemma 5. *The conditional probability (given* α, $\gcd(e, pq) = 1$, *and* $\{d_a : a \in A\}$ *such that* $e \not| \gcd\{d_a : a \in A\}$*) that* e *divides* $d = \sum_a v_a d_a$ *is at most* $2/3$.

We conclude the reduction showing that if $e > 0$ and $e \not| d$, then we can solve the strong QR-RSA problem (N, γ). The proof is based on the following theorem.

Theorem 3. *For any abelian group, there is a polynomial time algorithm that on input* (γ, ξ, e, d), *where* γ, ξ *are group elements and* e, d *integers, satisfying* $\xi^e = \gamma^d, e \neq 0, e \not| d$, *outputs* (θ, c, h) *such that* $\theta^{c^{h+1}} = \gamma^{c^h}$, $|c| \geq 2$, $c^{h+1} | e$ *and* $c^h | d$.

Proof. We define the algorithm $A(\gamma, \xi, e, d)$ recursively, by induction on the size of e and d. At each iteration, either d or e is replaced by a proper factor, while the other number is unchanged. It follows that the algorithm terminates within at most $\log_2(de)$ iterations.

Algorithm $A(\gamma, \xi, e, d)$ works as follows:

- If $d \not| e$, compute $d_1 = \gcd(e, d)$ using the extended Euclidean algorithm to find integers e', d' such that $d_1 = e \cdot e' + d \cdot d'$. Then invoke recursively $A(\gamma, \xi^{d'} \gamma^{e'}, e, d_1)$ and output the result.
- Otherwise, $d | e$, and we can compute $c = e/d$. If d is a power of c, i.e., $d = c^h$ for some integer h, return (ξ, c, h).
- If d is not a power of c, let h be the largest exponent such that $c^h | d$. Invoke recursively $A(\gamma, \xi^{d/c^h}, c^{h+1}, d)$ and return the result.

We need to prove that the algorithm is correct, and that either e or d decreases at every iteration.

First consider the case $d \not| e$. The input to the recursive call is given by $\xi_1 = \xi^{d'} \gamma^{e'}$, $e_1 = e$ and $d_1 = \gcd(e, d)$. Notice that

$$\xi_1^{e_1} = (\xi^{d'} \gamma^{e'})^e = (\xi^e)^{d'} \gamma^{ee'} = \gamma^{dd' + ee'} = \gamma^{d_1}.$$

Moreover, $e_1 = e \neq 0$, and $e_1 \not| d_1$, because otherwise $e = e_1 | d_1 | d$, contradicting $e \not| d$. So, the input to the recursive call is valid, i.e., it satisfies the assumptions in the theorem. In order to ensure termination, we need to check that d_1 properly divides d. Assume for contradiction $d = d_1$. From the definition of d_1, it follows that $d | e$, but this contradicts the condition $d \not| e$ tested by the algorithm.

Now assume $d | e$. Notice that $d \neq 0$, because otherwise $e | d$. So, the quotient $c = e/d$ is well defined. Moreover, $|c| > 1$ because $e \neq 0$, and $e \not| d$. If $d = c^h$, then the algorithm terminates with output $\theta = \xi$, c, h. Notice that

$$\theta^{c^{h+1}} = \xi^{dc} = \xi^e = \gamma^d = \gamma^{c^h},$$

i.e., the output is correct.

Finally, consider the case when $d | e$, but d is not a power of c. Notice that $d \neq 0$ because $e \not| d$. Since $|c| > 1$ and $d \neq 0$, the maximum $h = \max\{h : c^h | d\}$ is well defined, and d/c^h is an integer. This time, the algorithm is recursively invoked on input $\xi_1 = \xi^{d/c^h}$, $e_1 = c^{h+1}$ and $d_1 = d$. This input satisfies

$$\xi_1^{e_1} = \xi^{dc^{h+1}/c^h} = \xi^{dc} = \xi^e = \gamma^d = \gamma^{d_1}.$$

Moreover $e_1 = c^{h+1} \neq 0$ because $c \neq 0$. Also, $e_1 \not| d_1$, because otherwise $c^{h+1} = e_1 | d_1 = d$, contradicting the maximality of h. This time, we want to prove that e_1 properly divides e. Clearly, $e_1 = c^h c | dc = e$. Now, assume $e_1 = e$. Then, $c^h = e_1/c = e/c = d$, and d is a power of c. □

Applying Theorem 3 to equation $\xi^e = \gamma^d$, we get values $c \geq 2$, $h \geq 0$ and $\theta \in \mathbb{Z}_N^*$ such that $\theta^{c^{h+1}} = \gamma^{c^h}$. If $\theta^c = \gamma$, then (θ, c) is a solution to the strong QR-RSA problem. So, assume $(\theta^c/\gamma) \neq 1$, and assume also, without loss of generality that h is the smallest integer such that $(\theta^c/\gamma)^{c^h} = 1$. Let $\delta = (\theta^c/\gamma)^{c^{h-1}}$. We know that $\delta \neq 1$ and $\delta^c = 1$.

The following lemma shows that δ is a quadratic residue.

Lemma 6. *The value $\delta = (\theta^c/\gamma)^{c^{h-1}}$ is a quadratic residue.*

Apply Lemma 1 to quadratic residue δ. Since $\delta \neq 1$, either δ is a generator for QR_N, or $\gcd(\delta - 1, N) \in \{P, Q\}$. As before, if $g = \gcd(\delta - 1, N) \in \{P, Q\}$, we can compute $\phi(N) = (P-1)(Q-1) = (g-1)(N/g-1)$ and output the trivial strong QR-RSA solution $(\gamma, \phi(N) + 1)$

So, assume δ is a generator for QR_N. Since $\delta^c = 1$, it must be $pq = o(\delta) | c$. In particular, we also have $\gamma^c = 1$, and $(\gamma, c+1)$ is a solution to the strong QR-RSA problem.

This completes the proof that if the RSA group is not pseudo-free, then we can solve the strong RSA problem. □

4 Systems of Equations

The intuition behind the definition of pseudo-free group is that no polynomial time adversary can "prove" that the given computational group is not free. The kind of proofs implicit in Definition 2 consist of a single equation which is unsatisfiable over the free group, but satisfiable over the computational group. This choice is motivated by the fact that unsatisfiability of equations over free groups and satisfiability over computational groups can be efficiently demonstrated. (Specifically, unsatisfiability over free abelian groups is decidable in polynomial time, and satisfiability over arbitrary computational groups can be proved by giving a satisfying assignment.) An immediate extension that comes to mind is to consider systems of equations. Satisfiability for systems of equations is defined in the obvious way: a variable assignment satisfies a system of equations if it simultaneously satisfies all the equations in the system. As observed in [20], for the case of non abelian free groups, the results in [14] (see also [12–Lemma 3 and Corollary 2 and 3]) allow to combine systems of equations into a single equation. Specifically, the method is based on showing that the two equations $x = 1$ and $y = 1$ are equivalent to the single equation $x^2 a x^2 a^{-1} = (yby b^{-1})^2$, and it allows to transform any finite system of equations into a single equation with exactly the same set of solutions. Unfortunately, the same is not true for abelian groups, and the set of solutions of a system of equations cannot in general be

represented by a single equation. Consider for example the equations $x = 1$ and $y = 1$. The solution to this system is clearly unique. However, no single equation in two variables can have a unique solution. (Any bivariate equation has always either zero or infinitely many solutions over the free group.)

In this section we show that in the case of abelian groups, it is still possible to transform systems of equations into a single equation which is equivalent to the system, but in a weaker sense than having exactly the same set of solutions. The transformation maps any system of equations to a single equation whose solution set is a superset of the solutions to the system. However, if the system is unsatisfiable, then also the single equation is guaranteed to be unsatisfiable. This weaker notion of equivalence is enough to prove that Definition 2 is equivalent to the following seemingly stronger definition.

Definition 3. *A family of computational groups $\mathcal{G} = \{G_N\}_{N\in\mathcal{N}}$ is pseudo-free if for any set A of polynomial size $|A| = p(k)$ (where k is a security parameter), and probabilistic polynomial (in k) time algorithm $\mathcal{A}$, the following holds. Let $N \in \mathcal{N}(k)$ be a randomly chosen group index, and $\alpha\colon A \to G_N$ a function defining $|A|$ group elements chosen independently at random according to the computational group sampling procedure. Then, the probability that $\mathcal{A}(N, \alpha) = (\{E^i\}_{i\in I}, \xi)$ outputs an unsatisfiable system of equations $\{E^i\}_{i\in I}$ (over variables X and constants A) together with a solution $\xi\colon X \to G_N$ to $\{E^i_\alpha\}_{i\in I}$ over G_N, is negligible.*

The transformation from systems of equations to single equations is described in the following theorem.

Theorem 4. *There is a polynomial time algorithm that on input a system of equations $\{E^i\}_{i\in I}$ over constants A and variables X, outputs a single equation E over the same sets of constants A and variables X, such that the following holds.*

- *If $\{E^i\}_{i\in I}$ is unsatisfiable (over the free abelian group generated by A), then E is also unsatisfiable;*
- *For any computational group G and assignment $\alpha\colon A \to G$, any solution $\xi\colon X \to G$ to $\{E^i_\alpha\}_{i\in I}$ is also a solution to E_α.*

The proof of the theorem is based on elementary lattice techniques. For a detailed introduction to lattices and their computational complexity the reader is referred to [16]. Here we briefly recall the basic definitions and simple facts about lattices used in the proof of Theorem 4. For any matrix $\mathbf{M}$ with rational entries the lattice generated by a matrix $\mathbf{M} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n]$ is the set $\mathcal{L}(\mathbf{M}) = \{\sum_i x_i \boldsymbol{m}_i \colon x_i \in \mathbb{Z} \text{ for } i = 1, \ldots, n\}$ of all integer linear combinations of the columns of $\mathbf{M}$. There is a polynomial time algorithm that on input two rational matrices $\mathbf{M}$ and $\mathbf{M}'$, determines if $\mathcal{L}(\mathbf{M}) \subseteq \mathcal{L}(\mathbf{M}')$, and if not, finds a vector $\boldsymbol{u} \in \mathcal{L}(\mathbf{M}) \setminus \mathcal{L}(\mathbf{M}')$. The dual of a lattice $\mathcal{L}(\mathbf{M})$ is the set of all vectors $\boldsymbol{u}$ in the linear span of the columns of $\mathbf{M}$ that have integer scalar product with all lattice vectors in $\mathcal{L}(\mathbf{M})$. The dual of a lattice is a lattice, and the dual of

the dual of a lattice equals the original lattice. The dual of a lattice $\mathcal{L}(\mathbf{M})$ is denoted $\hat{\mathcal{L}}(\mathbf{M})$. Moreover, there is a polynomial time algorithm that on input a rational matrix $\mathbf{M}$ outputs a rational matrix $\mathbf{M}'$ such that $\mathcal{L}(\mathbf{M}') = \hat{\mathcal{L}}(\mathbf{M})$. It immediately follows from the definition of dual lattice that $\mathcal{L}(\mathbf{M})$ is a sub-lattice of $\mathcal{L}(\mathbf{M}')$ (i.e., $\mathcal{L}(\mathbf{M}) \subseteq \mathcal{L}(\mathbf{M}')$) if and only if $\hat{\mathcal{L}}(\mathbf{M}')$ is a sub-lattice of $\hat{\mathcal{L}}(\mathbf{M})$ (i.e., $\hat{\mathcal{L}}(\mathbf{M}') \subseteq \hat{\mathcal{L}}(\mathbf{M})$). We are now ready to prove Theorem 4.

Proof. Let $\{E^i\}_{i\in I}$ be a system of equations over the set of constant symbols A and variables X, and let $\sigma\colon X \to \mathcal{F}(A)$ be a generic variable assignment. Write each equation E^i and the assignment $\sigma(x)$ as

$$E^i\colon \prod_{x\in X} x^{e_{i,x}} = \prod_{a\in A} a^{d_{i,a}}$$
$$\sigma(x) = \prod_{a\in A} a^{s_{x,a}},$$

where the $e_{i,x}$, $d_{i,a}$ and $s_{x,a}$ are integers for all $i \in I$, $x \in X$ and $a \in A$. We use notation $e_{*,*}$ to denote the matrix with $|I|$ rows and $|X|$ columns with integer entries $(e_{i,x})_{i\in I, x\in X}$, and $e_{i,*}$ and $e_{*,x}$ to denote the rows and columns of matrix $e_{*,*}$. The matrices $d_{*,*}$, $s_{*,*}$ and vectors $d_{i,*}$, $d_{*,a}$, $s_{x,*}$, $s_{*,a}$ are defined similarly. Notice that σ is a solution to the system of equations over the free group if and only if

$$\sum_{x\in X} e_{i,x} s_{x,a} = d_{i,a}$$

for all $i \in I$ and $a \in A$, or, equivalently, in matrix notation, $e_{*,*}s_{*,*} = d_{*,*}$. So, the system of equations is solvable over the free group if and only if the integer lattice $\mathcal{L}(e_{*,*})$ contains $\mathcal{L}(d_{*,*})$ as a sub-lattice. Moreover, the two lattices satisfy $\mathcal{L}(e_{*,*}) \supseteq \mathcal{L}(d_{*,*})$ if and only if their duals satisfy the reverse inclusion $\hat{\mathcal{L}}(e_{*,*}) \subseteq \hat{\mathcal{L}}(d_{*,*})$. The inclusion $\hat{\mathcal{L}}(e_{*,*}) \subseteq \hat{\mathcal{L}}(d_{*,*})$ can be checked using standard techniques, and if it is not satisfied, one can efficiently find a vector $(u_i)_{i\in I} \in \hat{\mathcal{L}}(e_{*,*})$ such that $(u_i)_{i\in I} \notin \hat{\mathcal{L}}(d_{*,*})$.

If $\hat{\mathcal{L}}(e_{*,*}) \subseteq \hat{\mathcal{L}}(d_{*,*})$, then the system of equations $\{E^i\}_{i\in I}$ is satisfiable over the free group, and the algorithm can simply output an arbitrary equation $E = E^i$ from the system. Clearly, any solution to the system is also a solution to E. Moreover, the other condition in the theorem is vacuously satisfied because $\{E^i\}_{i\in I}$ is satisfiable over the free group.

So, let us assume that $\hat{\mathcal{L}}(e_{*,*}) \not\subseteq \hat{\mathcal{L}}(d_{*,*})$, and let $u_* = (u_i)_{i\in I}$ be a vector such that $(u_i)_{i\in I} \in \hat{\mathcal{L}}(e_{*,*}) \setminus \hat{\mathcal{L}}(d_{*,*})$. We know that $\sum_i u_i e_{i,x}$ is an integer for all $x \in X$ because u_* belongs to the dual lattice $\hat{\mathcal{L}}(e_{*,*})$. Moreover, since $\mathcal{L}(e_{*,*})$ is an integer lattice, all entries u_i are rational numbers. It follows that for any $a \in A$, $\sum_i u_i d_{i,a}$ is a rational number, but $\sum_i u_i d_{i,a}$ is not an integer for some $a \in A$. Let M be the smallest integer such that $M \cdot \sum_i u_i d_{i,a}$ is an integer for all $a \in A$. In other words, let M be the least common multiple of the denominators of the fractions $\sum_i u_i d_{i,a}$ for all $a \in A$. The output of the algorithm is the equation

$$E\colon \prod_{x\in X} x^{M\cdot\sum_i u_i e_{i,x}} = \prod_{a\in A} a^{M\cdot\sum_i u_i d_{i,a}}.$$

We need to show that this equation satisfies the two properties in the theorem.

Let $\alpha\colon A \to G_N$ and $\xi\colon X \to G_N$ be two assignments such that ξ is a solution to the system $\{E^i_\alpha\}_{i\in I}$ over computational group G_N, i.e., $\prod_{x\in X} \xi(x)^{e_{i,x}} = \prod_{a\in A} \alpha(a)^{d_{i,a}}$ for all $i \in I$. It follows that

$$\begin{aligned}
\xi\left(\prod_{x\in X} x^{M\cdot\sum_i u_i e_{i,x}}\right) &= \prod_{i\in I}\left(\prod_{x\in X} \xi(x)^{e_{i,x}}\right)^{Mu_i} \\
&= \prod_{i\in I}\left(\prod_{a\in A} \alpha(a)^{d_{i,a}}\right)^{Mu_i} \\
&= \alpha\left(\prod_{a\in A} a^{M\cdot\sum_i u_i d_{i,a}}\right),
\end{aligned}$$

i.e., ξ is also a solution to equation E_α. This proves the second property. For the first property, since the system is unsatisfiable, we need to prove that E is also unsatisfiable over the free group $\mathcal{F}(A)$. Assume for contradiction that E is satisfiable over the free group and let $\sigma(x) = \prod_{a\in A} a^{s_{x,a}}$ be a solution, i.e.,

$$\prod_{x\in X}\left(\prod_{a\in A} a^{s_{x,a}}\right)^{M\cdot\sum_i u_i e_{i,x}} = \prod_{a\in A} a^{M\cdot\sum_i u_i d_{i,a}}.$$

Since the group $\mathcal{F}(A)$ is free, this is true if and only if

$$M\sum_{x\in X} s_{x,a}\sum_{i\in I} u_i e_{i,x} = M\cdot\sum_{i\in I} u_i d_{i,a}$$

for all $a \in A$. Since $\sum_{x\in X} s_{x,a}\sum_{i\in I} u_i e_{i,x}$ is an integer, the left hand side of the last equation is a multiple of M. So, the right hand side is also a multiple of M, and $\sum_i u_i d_{i,a}$ is an integer for all $a \in A$. But this is a contradiction because by construction (namely, by the choice of $(u_i)_{i\in I}$) there exists an $a \in A$ such that $\sum_i u_i d_{i,a}$ is not an integer. □

Corollary 1. *A family of computational groups $\{G_N\}_{N\in\mathcal{N}}$ satisfies Definition 2 if and only if it satisfies Definition 3.*

Proof. If a group family is pseudo-free in the sense of Definition 3, then it satisfies Definition 2 as well because single equations are a special case of systems containing only one equation. Conversely, assume a group family does not satisfies Definition 3, i.e., there exists an adversary $\mathcal{A}$ that on input a group index $N \in \mathcal{N}$ and random assignment $\alpha\colon A \to G_N$, outputs an unsatisfiable system of equations $\{E_i\}_{i\in I}$ over constants A and variables X, together with a solution

$\xi: X \to G_N$ to the system over the computational group G_N. Then, using Theorem 4, $\mathcal{A}$ can be easily converted into an adversary $\mathcal{A}'$ contradicting Definition 2. Namely, on input group index $N \in \mathcal{N}$ and random assignment $\alpha : A \to G_N$, adversary $\mathcal{A}'$ invokes $\mathcal{A}$ on input (N, α) to get an unsatisfiable system of equations $\{E_i\}_{i \in I}$ together with a solution ξ over the computational group G_N. Finally, $\mathcal{A}'$ transforms $\{E_i\}_{i \in I}$ into a single equation E using Theorem 4, and outputs E, ξ. By Theorem 4, equation E is unsatisfiable over the free group, and ξ is a solution to E_α over G_N, proving that the group family does not satisfies Definition 2. □

The following corollary immediately follows from Theorem 2 and 1.

Corollary 2. *Let $\mathcal{N}$ be a distribution over safe prime products such that the strong RSA problem modulo $N \in \mathcal{N}$ is hard. Then the family of computational groups $\mathbb{Z}_N^*$ of invertible integers modulo $N \in \mathcal{N}$ (with the modular multiplication group operation, and uniform sampling procedure over QR_N) satisfies Definition 3, i.e., it is pseudo-free with respect to systems of equations.*

5 Conclusion

We have given the first example of provably secure pseudo-free group under standard cryptographic assumptions. In particular, we proved that the RSA group $\mathbb{Z}_N^*$ where N is the product of two safe primes is pseudo-free, assuming the hardness of the strong RSA problem. Many open problems remain. In this section we illustrate some of them.

Our proof uses the fact that N is the product of two safe primes, and elements are sampled uniformly at random from the subgroup QR_N of quadratic residues. A natural question is whether $\mathbb{Z}_N^*$ is pseudo-free even when N is the product of two arbitrary primes, and elements are sampled uniformly at random from the whole group $\mathbb{Z}_N^*$. Another open problem is to relax the hypothesis of Theorem 2, and prove that $\mathbb{Z}_N^*$ is pseudo-free assuming that factoring N is hard. Notice that this last problem is probably very hard, as it would imply that inverting the RSA function is at least as hard as factoring, a long standing open problem in cryptography. However, there are many other cryptographic problems that have been proved at least as hard as factoring, like the discrete logarithm problem [2], the Diffie-Hellman problem [15], and the generalized Diffie-Hellman problem [5] modulo Blum integers. We remark that while computing discrete logarithms in pseudo-free groups is provably hard [20], no relation between pseudo-freeness and the Diffie-Hellman problem is currently known. An interesting open question, already posed in [20], is to show that the Diffie-Hellman problem in pseudo-free groups is computationally hard.

Another interesting problem is to find other examples of pseudo-free groups, beside $\mathbb{Z}_N^*$, and possibly proving their security based on standard cryptographic assumptions. Of particular interest would be to find a good candidate of non abelian pseudo-free group.

Finally, it would be nice to find applications of pseudo-free groups, as those mentioned in [20] and in the introduction, to demonstrate the usefulness of the

notion of pseudo-free group. It might be the case that some applications require even stronger notions of pseudo-freeness than the one defined in [20]. In Section 4 we already considered extending the definition to systems of equations, and proved that pseudo-freeness with respect to systems of equations (Definition 3) is equivalent to the basic definition of pseudo-free group. Another possible extensions is to consider more general boolean combinations of equations, e.g., one can consider systems of equations $w_1 = w_2$ and inequations $w_1 \neq w_2$. For example, $x^2 = 1$ and $x \neq 1$ cannot be simultaneously satisfied over the free group, but admit a solution $x = N - 1$ in $\mathbb{Z}_N^*$ for any $N \neq 2$. We remark that the satisfiability problem over free abelian groups for arbitrary boolean combinations of equations is NP-hard. (E.g., 3SAT can be immediately reduced to such a formula mapping each boolean variable x to a corresponding equation $x = 1$.) So, some unsatisfiable formula do not have short (polynomial size) proofs of unsatisfiability, unless NP=coNP. Extensions of the notion of pseudo-free group to general boolean combinations of formulas should require the adversary to output not only an unsatisfiable formula over the free group (together with a solution over the computational group), but also a short and easily verifiable proof that the formula is indeed unsatisfiable.

Acknowledgments

Research supported in part by NSF grants 0093029, 0313241, 0430595 and a Sloan research fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. M. Abadi and P. Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.
2. E. Bach. Discrete logarithms and factoring. Technical Report CSD-84-186, University of California at Berkeley, 1984.
3. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proceedings of the 10th ACM conference on computer and communications security - CCS 2003*, pages 220–230, Washington, DC, USA, Oct. 2003. ACM.
4. N. Baric' and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in cryptology - EUROCRYPT '97, Proceedings of the international conference on the theory and application of cryptographic techniques*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494, Konstanz, Germany, May 1997. Springer.
5. E. Biham, D. Boneh, and O. Reingold. Breaking generalized Diffie-Hellman modulo a composite is no easier than factoring. *Information Processing Letters*, 70(2):83–87, Apr. 1999.

6. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000. Preliminary version in CCS 1999.
7. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
8. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In B. S. Kaliski, Jr., editor, *Advances in cryptology - CRYPTO '97, Proceedings of the 17th annual international cryptology conference*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30, Santa Barbara, California, USA, Aug. 1997. Springer.
9. R. Gennaro, T. Rabin, and H. Krawczyk. RSA-based undeniable signatures. *Journal of Cryptology*, 13(4):397–416, 2000. Preliminary version in Crypto 1997.
10. S. Hohenberger. The cryptographic impact of groups with infeasible inversion. Master's thesis, Massachusetts Institute of Technology, EECS Dept., Cambridge, MA, June 2003.
11. R. Impagliazzo and B. Kapron. Logics for reasoning about cryptographic constructions. In *Proceedings of the 44rd annual symposium on foundations of computer science - FOCS 2003*, pages 372–383, Cambridge, MA, USA, Nov. 2003. IEEE.
12. O. Kharlampovich and A. Myasnikov. Implicit function theorem over free groups. *Journal of Algebra*, 2005. To appear.
13. P. D. Lincoln, J. C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic polytime framework for protocol analysis. In *Proceedings of the fifth ACM conference on computer and communications security - CCS '98*, pages 112–121, San Francisco, California, USA, Nov. 1998. ACM.
14. A. I. Mal'cev. On some correspondence between rings and groups. *Math. Sbornik*, 50:257–266, 1960.
15. K. S. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology*, 1(2):95–105, 1988.
16. D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.
17. D. Micciancio and S. Panjwani. Adaptive security of symbolic encryption. In J. Kilian, editor, *Theory of cryptography conference - Proceedings of TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 169–187, Cambridge, MA, USA, Feb. 2005. Springer.
18. D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004. Preliminary version in WITS 2002.
19. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In M. Naor, editor, *Theory of cryptography conference - Proceedings of TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, Cambridge, MA, USA, Feb. 2004. Springer.
20. R. L. Rivest. On the notion of pseudo-free groups. In M. Naor, editor, *Theory of cryptography conference - Proceedings of TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 505–521, Cambridge, MA, USA, Feb. 2003. Springer.
21. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

Universally Composable Password-Based Key Exchange

Ran Canetti[1,⋆], Shai Halevi[1], Jonathan Katz[2,⋆⋆], Yehuda Lindell[3,⋆⋆⋆], and Phil MacKenzie[4,†]

[1] IBM T.J. Watson Research Center, Hawthorne, NY, USA
canetti@watson.ibm.com, shaih@alum.mit.edu
[2] Dept. of Computer Science, University of Maryland, MD, USA
jkatz@cs.umd.edu
[3] Department of Computer Science, Bar-Ilan University, Israel
lindell@cs.biu.ac.il
[4] Bell Labs, Lucent Technologies, Murray Hill, NJ, USA

Abstract. We propose and realize a definition of security for password-based key exchange within the framework of universally composable (UC) security, thus providing security guarantees under arbitrary composition with other protocols. In addition, our definition captures some aspects of the problem that were not adequately addressed by most prior notions. For instance, it does not assume any underlying probability distribution on passwords, nor does it assume independence between passwords chosen by different parties. We also formulate a definition of password-based secure channels, and show that such a definition is achievable given password-based key exchange.

Our protocol realizing the new definition of password-based key exchange is in the common reference string model and relies on standard number-theoretic assumptions. The components of our protocol can be instantiated to give a relatively efficient solution which is conceivably usable in practice. We also show that it is impossible to satisfy our definition in the "plain" model (e.g., without a common reference string).

1 Introduction

Protocols for password-based key exchange have received much attention in recent years. In short, the problem is how to enable authenticated generation of a "high-quality" secret key between two parties whose only *a priori* shared, secret information consists of a low-entropy password. In this setting, an attacker can always correctly determine the correct password via an *on-line dictionary attack*

⋆ Supported by NSF CyberTrust Grant 0430450.
⋆⋆ Supported by NSF CAREER award 0447075 and Trusted Computing grant 0310751.
⋆⋆⋆ Some of this work was carried out while the author was at IBM T.J. Watson.
† Current addr.: DoCoMo USA Labs, San Jose, CA. (philmac@docomolabs-usa.com)

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 404–421, 2005.

in which the adversary exhaustively enumerates the password space and tries to impersonate one of the parties using each possible shared secret. Since such an attack is unavoidable, work in this area focuses on preventing *off-line dictionary attacks*. Roughly, this guarantees that the exhaustive, on-line attack is the "best" possible one. That is, the attacker must interact with a legitimate player in order to verify each password guess, and the interaction leaks no information other than whether or not the attacker's guess is correct. Besides their practical importance, password-based protocols are also interesting from a purely theoretical point of view: they provide a rare case where bootstrapping "strong security" from "weak security" has to be modeled, obtained, and argued.

The problem of resistance to off-line password-guessing attacks was first raised by Gong, et al. [21] in the asymmetric "PKI model" (where, in addition to a password, the user has the public key of the server). Formal definitions and proofs of security in this setting were later given by Halevi and Krawczyk [22]. A more difficult setting for this problem is one where the parties share *only* a password (and in particular, neither party knows the other's public-key). This setting was first considered by Bellovin and Merritt [5], and their work was followed by much additional research developing protocols with heuristic justifications for their security (see [6] for a survey). Formal definitions for this setting, together with protocols analyzed in the random-oracle/ideal-cipher models, were given by Bellare, et al. [3] (who proposed an indistinguishability-based definition) and Boyko, et al. [7] (who proposed a simulation-based definition). Goldreich and Lindell [19] introduced a third security definition and also gave the first provably-secure solution to this problem in the standard model, based on general assumptions; their protocol was recently simplified (at the expense of achieving a weaker security guarantee) by Nguyen and Vadhan [26]. Another setting that has been considered for this problem is one where, in addition to shared low-entropy passwords, all parties share a common reference string. In this setting, a practical and provably-secure protocol was first developed by Katz, et al. [24] based on the decisional Diffie-Hellman assumption. This protocol was subsequently generalized and abstracted by Gennaro and Lindell [18] who, among other things, obtain protocols that rely on the quadratic residuosity and N^{th}-residuosity assumptions.

The many definitions that have already been introduced [3, 7, 19, 26] indicate that finding a "good" definition of security for password-based authentication has been difficult and remains a challenging problem. Furthermore, it is not clear that any of the above definitions adequately address all aspects of the problem. For example, none of the above definitions relate to the (realistic) setting where the password-based protocol is used as a component within a larger protocol. (Rather, it is assumed that the entire network activity consists of many executions of the password protocol.) Since the problem at hand involves *non-negligible* probabilities of "success" by the adversary, providing security-preserving composition (with reasonable error propagation) is even more delicate than usual. Some of the above definitions have not been proven sufficient for implementing (any form of) secure channels — a natural goal of key-exchange protocols (see [9] for motivation). Finally, existing (explicit) definitions assume

that passwords are chosen from some *pre-determined, known distribution*, and (with the exception of [7]) assume also that passwords shared between different parties are *independent*. (However, it is claimed in [24] that their proof extends to the case of dependent passwords.) These assumptions rarely hold in practice.

A new definition. In this work, we propose and realize a new definition of security for password-based key-exchange protocols within the universally composable (UC) security framework [8]. That is, we propose an ideal functionality for "password-based key exchange" that captures the security requirements of the problem. (Such an ideal functionality can be thought of as the code for a "centralized trusted service", were one actually available to the parties.) Working in the UC framework allows us to benefit from the universal composition theorem. Loosely speaking, the theorem states that a protocol secure in this framework remains secure even when run in an arbitrary network, where many different protocols (secure or not) may run concurrently. In addition to addressing composability, the definition in this work also addresses the other concerns mentioned above. In particular, security is preserved even in the case of arbitrary and unknown password distributions, and even if related passwords are used. The important feature here is that the probability of the adversary succeeding in its attack is negligibly close to the probability of its guessing the password outright, even when this guess is based on information about the password that the adversary obtains from the arbitrary network or from related passwords that it has (either partially or completely) learned. Finally, we show how protocols satisfying our definition may be used to construct (password-based) secure channels. Such channels enable private and authenticated communication, which in most cases is the goal of running the protocol in the first place.

As one might expect, formulating an ideal functionality that captures all the requirements of password-based key exchange involves a number of non-trivial definitional choices. Our formulation builds on the known UC formulation of (standard) key-exchange [8, 9], where security is guaranteed except with negligible probability. However, unlike standard key-exchange, some mechanism must be introduced that allows the adversary to "break" the protocols with some non-negligible probability by guessing the correct password. A natural way of doing this is to have the functionality choose the passwords for the parties. Then, if the adversary correctly guesses the password (where this guess is given to the functionality), it is allowed to choose the session-key that the parties obtain. Although this formulation is quite intuitive, it is somewhat limited in that it assumes some pre-determined dictionary or distribution on passwords and that passwords are chosen independently from each other. It also fails to model possible leakage of a portion of the password to the adversary (this is due to the fact that only the functionality knows the password).

We therefore take a different approach and allow the calling protocol (or the environment) to provide the password as part of the input. While this formulation may seem somewhat counter-intuitive at first, we show that it results in a definition of security that is at least as strong as that given by the first formu-

lation.[1] Furthermore, it does not make any assumptions as to how the password is chosen and it imposes no pre-determined probabilities of failure.

Realizing the definition. We construct a protocol that realizes our definition. The protocol is an extension of the protocols of [24, 18], and as such is in the common reference string model and may be based on some standard number-theoretic assumptions (namely the decisional Diffie-Hellman, quadratic residuosity, or N^{th}-residuosity assumptions). Our protocol uses building blocks that have efficient instantiations under these assumptions. As a result, our protocol is reasonably efficient and is realizable in practice (it has 6 rounds and at most 30 modular exponentiations per party). Some of the efficiency improvements we use in our protocol are applicable also to the protocol of [24] (and seemingly [18]); see [23]. Applied there, these improvements yield the most efficient known password-based protocol meeting the definition of [3] without random oracles.

On the necessity of set-up assumptions. Our protocol is constructed in the common reference string model, and so requires a trusted setup phase. In fact, we show that our UC-based definition of password-based key-exchange cannot be securely realized by any protocol in the *plain model* (i.e., in a model with no trusted setup whatsoever). Beyond providing some justification for our use of a common reference string, this result stands in sharp contrast with the fact that standard UC-secure key exchange *can* be realized in the plain model [9]. It also shows that our definition is strictly stronger than the definition used by [19, 26], which *can* be realized in the plain model. (We stress that in contrast to our definition, the definitions of [19, 26] do not guarantee security even under concurrent composition of the same protocol with the same password.)

Password-based secure channels. Perhaps the most important application of key-exchange protocols is for establishing secure communication sessions between pairs of parties. To advocate the adequacy of our proposed definition we formulate a UC notion of password-based secure channels, and show how to realize it given our notion of password-based key exchange. It is of course impossible to obtain standard secure channels using short passwords, since the adversary may guess the password with non-negligible probability. Consequently, our notion of password-based secure channels relaxes the standard notion in a way similar to which our notion of password-based key exchange relaxes the standard notion of key exchange. We then show that the standard protocols for realizing secure channels based on standard key exchange (see, e.g., [9]), suffice also for realizing password-based secure channels from password-based key exchange.

Organization. Due to lack of space in this abstract, the main text focuses on our definition and a high-level description of our protocol, along with motivation as to its security. The full description of the protocol and its proof of security is

[1] The alternative formulation in which the functionality chooses the passwords was omitted for lack of space. It may be obtained from the authors, along with a proof that it is implied by the definition presented here.

provided in the full version. Other results (namely, the impossibility of realizing our definition in the plain model and the fact that secure channels are implied) are described briefly in Section 4 and can be found in the full version.

2 Definitions of Security

In this section, we motivate and present our formulation of an ideal functionality for password-based key exchange in the UC framework. We stress that from here on, when we say that a protocol securely realizes some functionality, we mean that it securely realizes it according to the definitions of the UC framework. Our presentation assumes familiarity the UC framework; see [8] for a full description.

2.1 High-Level Approach

The starting point for our approach is the definition for universally composable "standard" key-exchange [9] (cf. Figure 1). Our aim is to define a functionality that achieves the same effect as standard key-exchange (where the parties have high-entropy keys), except that we also incorporate the inherent "security defect" due to the use of low-entropy passwords. Two ways of introducing this "security defect" come to mind:

1. One option is to consider the same functionality $\mathcal{F}_{\mathsf{KE}}$ as in Figure 1, but to relax the requirement of indistinguishability between the real and ideal worlds. I.e., when passwords are assumed to be chosen uniformly from a dictionary $\mathcal{D}$, one would define a secure protocol as one whose real-world execution is distinguishable from an interaction with the ideal functionality with probability at most, say, $1/|\mathcal{D}|$ plus a negligible amount.
2. A second possibility is to incorporate the "defect" directly into the functionality, e.g., by allowing the adversary to make explicit password guesses and to "break" the protocol following a successful guess. Here, the adversary "breaks" the protocol with noticeable probability even in the ideal world, and thus the standard notion of realizing an ideal functionality can be used.

Among previous works that used simulation-based definitions of security for password protocols, the first approach was taken by [19, 26], while the second was taken by [7]. (Other definitions are not simulation-based and so do not fit into either approach.) In this work we adopt the second option, for two reasons. First, this allows us to use the UC composition theorem and thus guarantee security of password-based key-exchange protocols even when run in arbitrary protocol environments. Second, this approach easily extends to handle additional complexities such as multiple users with different distributions on their passwords, or dependencies among various passwords. These aspects seem hard (if not impossible) to handle using the first approach.

Before proceeding to our definition, we describe the standard key-exchange functionality of [9]. (We note that the formulation of $\mathcal{F}_{\mathsf{KE}}$ in Figure 1 is somewhat different from the one in [9]; however, the differences are inconsequential for the purpose of this work.) The main idea behind the $\mathcal{F}_{\mathsf{KE}}$ functionality is as

follows: If both participating parties are not corrupted, then they receive the same uniformly distributed session-key, and the adversary learns nothing of the key except that it was generated. However, if one of the parties *is* corrupted, then the adversary is given the power to fully determine the session-key. The rationale for this is that the aim of key-exchange is to enable honest parties to generate a key that is *unknown* to an external adversary. If one of the participating parties is corrupted, then the adversary will learn the generated key (because it is one of the participants), and so the security requirement is meaningless. In such a case, there is nothing lost by allowing the adversary to determine the key. We remark that the "role" variable in the NewSession message is included in order to let a party know if it is playing the initiator or responder role in the protocol. This has no effect on the security, but is needed for correct executions.

Functionality $\mathcal{F}_{\mathsf{KE}}$

$\mathcal{F}_{\mathsf{KE}}$ is parameterized by a security parameter k. It interacts with an adversary S and a set of (dummy) parties via the following queries.

Upon receiving a query $(\mathsf{NewSession}, sid, P_i, P_j, \text{role})$ **from party** P_i**:**
Send $(\mathsf{NewSession}, sid, P_i, P_j, \text{role})$ to S. In addition, if this is the first NewSession query, or if this is the second NewSession query and there is a record (P_j, P_i), then record (P_i, P_j).

Upon receiving a query $(\mathsf{NewKey}, sid, P_i, sk)$ **from** S**, where** $|sk| = k$**:**
If there is a record (P_i, P_j), and this is the first NewKey query for P_i, then:
- If either P_i or P_j is corrupted, then output (sid, sk) to player P_i.
- If there is also a record (P_j, P_i), and a key sk' was sent to P_j, output (sid, sk') to P_i.
- In any other case, pick a new random key sk' of length k and send (sid, sk') to P_i.

Fig. 1. The authenticated key-exchange functionality $\mathcal{F}_{\mathsf{KE}}$

We now proceed to our definition of the password-based key-exchange functionality $\mathcal{F}_{\mathsf{pwKE}}$. Similarly to $\mathcal{F}_{\mathsf{KE}}$, if one of the participating parties is corrupted the adversary is given the power to fully determine the resulting session-key. However, this power is also given to the adversary in case it succeeds in guessing the parties' shared password. An additional property of our definition is that *failed* adversarial attempts at guessing a key are detected by the participating parties. Specifically, if the adversary makes a wrong password guess in a given session, then the session is marked interrupted and the parties are provided independently-chosen session-keys. (Giving the parties error messages in this case would correspond to requiring explicit mutual authentication; see additional discussion below.)

In the functionality, a session is marked compromised if the adversary makes a successful password guess (as discussed above, in this case the adversary is allowed to determine the session-key). If a session is marked fresh, this means that it is neither interrupted nor compromised. Such sessions (between honest parties) conclude with both parties receiving the same, uniformly distributed session-key. See Figure 2 for the full definition of the functionality.

In the definition of $\mathcal{F}_{\mathsf{pwKE}}$, the password is chosen by the environment who then hands it to the parties as input.[2] Since we quantify over all (polynomial-time) environments, this implies that security is preserved for all efficient password distributions, as well as when arbitrarily related passwords are used in different session. Furthermore, since the passwords are provided by the "environment in which the protocol is run", security is preserved even when passwords are used for other, unintended purposes by that same environment. (When we say that security is preserved here, we mean that the probability that an adversary can break the password-based key-exchange protocol is the same as its probability of guessing the password outright, given the potential misuse mentioned above.) We also remark that our definition guarantees security even in the case where two honest players execute the protocol with different passwords. (In fact, this is quite a realistic scenario which occurs every time a user mistypes a password; previous definitions did not guarantee anything in such a case.)

Functionality $\mathcal{F}_{\mathsf{pwKE}}$

The functionality $\mathcal{F}_{\mathsf{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary S and a set of parties via the following queries.

Upon receiving a query $(\mathsf{NewSession}, sid, P_i, P_j, pw, \text{role})$ **from party** P_i**:** Send $(\mathsf{NewSession}, sid, P_i, P_j, \text{role})$ to S. In addition, if this is the first NewSession query, or if this is the second NewSession query and there is a record (P_j, P_i, pw'), then record (P_i, P_j, pw) and mark this record fresh.

Upon receiving a query $(\mathsf{TestPwd}, sid, P_i, pw')$ **from the adversary** S**:** If there is a record of the form (P_i, P_j, pw) which is fresh, then do: If $pw = pw'$, then mark the record compromised and reply to S with "correct guess". If $pw \neq pw'$, then mark the record interrupted and reply with "wrong guess".

Upon receiving a query $(\mathsf{NewKey}, sid, P_i, sk)$ **from** S**, where** $|sk| = k$**:** If there is a record of the form (P_i, P_j, pw), and this is the first NewKey query for P_i, then:

- If this record is compromised, or either P_i or P_j is corrupted, then output (sid, sk) to player P_i.
- If this record is fresh, and there is a record (P_j, P_i, pw') with $pw' = pw$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was fresh at the time, then output (sid, sk') to P_i.
- In any other case, pick a new random key sk' of length k and send (sid, sk') to P_i.

Fig. 2. The password-based key-exchange functionality, $\mathcal{F}_{\mathsf{pwKE}}$

As additional justification of our definition, we show that it implies password-based secure channels (arguably the most common application of such protocols).

[2] This is in contrast to an alternative approach described in the Introduction where the functionality chooses the password according to some predetermined distribution, and this password is hidden even from the environment. As we have mentioned, security under our definition implies security under that alternative approach.

In addition, we show that our definition implies the "expected" notion of security against a passive eavesdropper (even one who happens to know the password being used). Finally, we show that a protocol that securely realizes our functionality is secure also with respect to the definition of Bellare, et al. [3] (modulo unimportant differences regarding the formalization of session identifiers).[3] These last two results can be seen as "sanity checks" of our definition.

Additional discussion. The definition of $\mathcal{F}_{\mathsf{pwKE}}$ could be *strengthened* to require explicit mutual authentication by insisting that after a "wrong guess" of the password, the session would fail (instead of producing a random and independent key). Similarly, a session with mismatching passwords would also fail. We chose not to include these requirements because **(a)** we want to keep the exposition simple; **(b)** mutual authentication is not needed for secure channels; and **(c)** it is well known that any secure key-exchange protocol (including ours) can be augmented to provide mutual authentication by adding two "key confirmation" flows at the end (and refreshing the session key). The definition could also be *weakened* by notifying the simulator whether or not the passwords match in the two NewSession queries. Roughly, the difference is that the current formulation requires that an eavesdropper be unable to detect whether the session succeeded (i.e., both parties got the same key) or failed (i.e., they got different keys). Although we are not aware of any application where this is needed, it makes the definition simpler to describe and our protocol anyway satisfies this requirement.

3 A Protocol Securely Realizing $\mathcal{F}_{\mathsf{pwKE}}$

In this section, we present our protocol for securely realizing the functionality $\mathcal{F}_{\mathsf{pwKE}}$, in the common reference string model.[4] Due to lack of space in this extended abstract, many details of the protocol and its proof of security are omitted. We remark that our protocol is proven secure in the model of *static corruptions* (where the adversary may corrupt some of the participants, but only prior to the beginning of the protocol executions) and *unauthenticated channels* (where the adversary has full control over the communication channels and can insert, modify and delete all messages sent by the honest parties). We also note that although we consider static corruptions, the "weak-corruption model" of [3] is implied by our definition (and achieved by our protocol); see the full version. In the weak-corruption model, the adversary may obtain passwords adaptively

[3] In our formalization, a unique session identifier *sid* is assumed to be part of the input to the functionality. In the two-party setting, such a session identifier can be obtained by having the parties exchange random strings and then set *sid* to be the concatenation of these strings.

[4] Our protocol actually realizes the multi-session extension $\hat{\mathcal{F}}_{\mathsf{pwKE}}$ of this functionality (see [11]). This is important for ensuring that the same common reference string can be used in all executions; see the full version for more details. For the sake of clarity, in this extended abstract we refer only to the original functionality $\mathcal{F}_{\mathsf{pwKE}}$.

throughout the execution. This essentially models leakage of passwords, rather than adaptive corruption of parties.

3.1 Preliminaries

The protocol uses a number of primitives: one-time signatures, CPA-secure and CCA-secure public-key encryption, simulation-sound zero-knowledge proofs, and smooth projective hashing. We provide only a brief description of the last two primitives here.

Simulation-sound zero-knowledge (SSZK) proofs. Informally speaking, a zero-knowledge proof system is said to be (unbounded) *simulation-sound* if it has the property that an adversary cannot provide a convincing proof for a false statement, even if it has seen *simulated proofs*. Such simulated proofs may actually prove false statements, and so the adversary can copy these proofs but do nothing more. More formally, the adversary is given oracle access to the zero-knowledge simulator and can request simulated proofs of any statement (true or false) that it wishes. The adversary is then said to *succeed* if it generates a convincing proof of a false statement, and this proof was not received from the oracle. This concept was first introduced by Sahai [28] and De Santis, et al. [14] in the context of non-interactive zero-knowledge. For the case of interactive protocols, the notion was formally defined by Garay, et al. [17].[5] Efficient methods for transforming three-round honest-verifier zero-knowledge protocols (also called Σ-protocols [12]) into simulation-sound zero-knowledge protocols in the common reference string model have been shown in [17] and [25]. We note that, according to the definition of [17], simulation-sound zero knowledge protocols also achieve *concurrent* zero knowledge; i.e., the zero knowledge property holds for an unbounded number of asynchronous executions of an honest prover. Finally, we note that simulation-sound zero knowledge is a weaker requirement than universally-composable zero-knowledge, and more efficient constructions for it are known.

Smooth projective hashing [13]. On a very high level, a projective hash family is a family of hash functions that can be computed using one of two keys: the (secret) hashing key can be used to compute the function on every point in its domain, whereas the (public) projected key can only be used to compute the function on a specified subset of the domain. Such a family is called "smooth" if the value of the function on a value outside of the specified subset is uniformly distributed, even given the projected key. More formally (but still far from being exact), let X be a set and let $L \subset X$. We say that a hash function H_{hk} that maps X to some set is projective if there exists a projection function $\alpha(\cdot)$ that maps hash keys hk into their projections $hp = \alpha(hk)$, such that for every $x \in L$ it holds that the value of $H_{hk}(x)$ is uniquely determined by hp and x. (In contrast, for $x \notin L$ the value of $H_{hk}(x)$ need not be determined from hp and x.) A

[5] When we say a proof is simulation sound, we will also mean that it is *uniquely applicable* [28]; that is, a proof is valid for at most one statement.

smooth projective hash function has the additional property that for $x \notin L$, hp actually says *nothing* about the value of $H_{hk}(x)$. More specifically, given x and $hp = \alpha(hk)$, the value $H_{hk}(x)$ is (statistically close to) a uniformly distributed element in the range of H_{hk}.

We already mentioned that for $x \in L$ the projected key hp fully defines the value $H_{hk}(x)$, but so far we said nothing about whether or not this value can be efficiently computed. An important property of smooth projective hash functions is that if the subset L is *an NP-language*, then for $x \in L$ it is possible to compute $H_{hk}(x)$ using the projected key $hp = \alpha(hk)$ and a witness of the fact that $x \in L$. Thus, for $x \in L$ there are two alternative ways of computing $H_{hk}(x)$:

1. Given the hashing key hk, compute $H_{hk}(x)$ directly.
2. Given the projected key hp and a witness w for $x \in L$, compute the hash value $h_{hp}(x; w) = H_{hk}(x)$.

Following [18], the set X that we consider in this work is the set $\{(c, m)\}$ of all ciphertext/plaintext pairs under a given public-key pke. Furthermore, the language L is taken to be $\{(c, m) \mid c = E_{pke}(m)\}$; that is, L is the set of all ciphertext/plaintext pairs (c, m) where c is an encryption of m under the public-key pke. We note this language is indeed an NP language, with the witness being the randomness that was used in the encryption of m.

We also comment that the semantic security of the encryption implies that L is hard on the average (i.e., it is hard to distinguish a random element in X from a random element in L). For such languages, it was proven in [18] that given a random $x \in L$ and $hp = \alpha(hk)$, the value $H_{hk}(x)$ is *computationally indistinguishable* from a random value in the range of H_{hk}. (This holds even though for any $x \in L$, the value $H_{hk}(x)$ is uniquely determined by x and hp.)

In the description below we denote choosing a hashing key from the family by $hk \leftarrow \mathcal{H}$, and denote the projection of this key by $hp = \alpha(hk)$. We also denote computing the hash value using the hashing key hk by $H_{hk}(x)$, and computing the hash value using the projected key hp and witness w by $h_{hp}(x; w)$. (Note that the statements x below are actually pairs (c, m), and the witness is the randomness r, so we write $H_{hk}(c, m)$ and $h_{hp}(c, m; r)$.)

3.2 The KOY/GL Protocol

The starting point of our protocol is the password-based key-exchange protocol of Katz, Ostrovsky, and Yung [24], as generalized and abstracted by Gennaro and Lindell [18]. The "core" of this protocol is sketched in Figure 3 (in this figure we suppress various details). At a high level, the parties in the KOY/GL protocol exchange CCA-secure encryptions[6] of the password, encrypted with the public-key found in the common reference string, and then compute the session key by combining (smooth projective) hashes of the two ciphertext/password pairs. In

[6] It is shown in [18] that non-malleable commitments can be used in place of CCA-secure encryption. However, for our extension of the protocol to the UC framework we will need to use encryption, and so we describe it in this way.

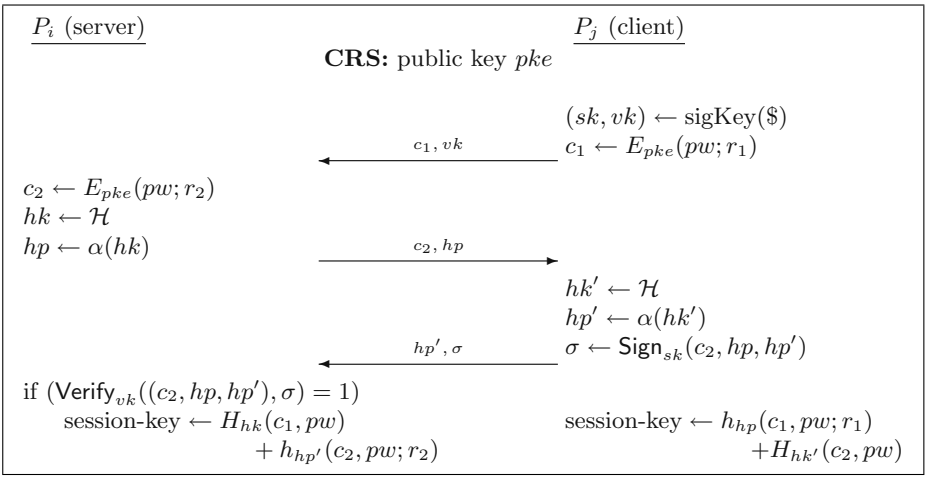

Fig. 3. The core of the KOY/GL protocol

order to do this, each party chooses a hashing key for a smooth projective hash function and sends the "projected key" to the other party.

Ignoring for the moment the signature keys from Figure 3, let c_2, hk and c_1, hk' be the encryptions and hashing keys generated by parties P_i and P_j, respectively. Party P_i can compute $H_{hk}(c_1, pw)$ since it knows the actual hashing key hk. Furthermore, since it generated the ciphertext c_2, it can compute $h_{hp'}(c_2, pw; r_2) = H_{hk'}(c_2, pw)$ using its knowledge of the randomness r_2 that was used in generating $c_2 = E_{pke}(pw; r_2)$. (This relies on the two alternative ways of computing $H_{hk}(x)$ described above.) Symmetrically, P_j computes the same session key using hk', hp, and its knowledge of r_1.

The basic idea behind the security of the protocol can be described as follows. Denote the shared password of a client and server by pw. If the client receives an encryption c of the wrong password pw', then (by the definition of smooth projective hashing) the hash she computes will be random and independent of all her communication. (This holds because the statement (c, pw) is not in the language, so $H_{hk}(c, pw)$ is close to uniform even given the projected key hp.) A similar argument holds for the server. Thus, for an adversary to distinguish a session key from random, it must send one of the parties an encryption of the *correct* password pw.

The adversary can get an encryption of the right password by copying a ciphertext from another execution of the protocol, but then it does not know the randomness that was used to generate this ciphertext. By the property that we discussed above (regarding hard-on-the-average languages), the value $H_{hk}(c, pw)$ is computationally indistinguishable from uniform, even given hp. Moreover, since the encryption scheme is CCA-secure, and thus non-malleable, the adversary cannot generate a new encryption of pw with probability any better than it would achieve by simply guessing passwords from the dictionary and encrypting them.

Finally, the adversary may try to gain information by copying ciphertexts from a current session faithfully but not copying other values (such as the hash projected keys). This type of man-in-the-middle attack is prevented using the one-time signature. We conclude that the adversary succeeds in its attack if and only if it generates an encryption of the correct password. In other words, the adversary succeeds if it guesses the secret password, as required.

3.3 Extending the Protocol to Realize $\mathcal{F}_{\mathsf{pwKE}}$

The protocol of Figure 3 serves as a good starting point, but it does not seem to achieve the security that we require. The main issue that arises here is that an ideal-model simulator must be able to extract the adversary's password guess.[7] At first glance, it may seem that this is not a problem because in the ideal model the simulator has control over the common reference string and so can include a public-key pke for which it knows the corresponding secret key ske. Then, when the adversary generates an encryption of the password $c = E_{pke}(pw)$, the simulator can decrypt using ske and obtain the password guess pw. However, as we will now show, this seems not to suffice.

In order to see where the difficulty arises, consider an ideal-model adversary/simulator $\mathcal{S}$ that has access to the functionality $\mathcal{F}_{\mathsf{pwKE}}$ and needs to simulate the KOY/GL protocol for a real-life adversary $\mathcal{A}$. Informally, simulating the server when the adversary impersonates a *client* can be carried out as follows: The simulator decrypts the ciphertext c_1 generated by the adversary and recovers the adversary's "password guess" pw (this decryption can be carried out because $\mathcal{S}$ chooses the common reference string so that it has the corresponding secret key). The simulator then sends pw to $\mathcal{F}_{\mathsf{pwKE}}$ as its own guess. If the guess is incorrect, then as described above, the smoothness of the hash function causes the honest parties to output independent random keys (as required in the ideal model with an $\mathsf{interrupted}$ session). In contrast, if the guess is correct then the simulator has learned the correct password and can continue the remainder of the execution exactly as an honest party would when using that password. However, consider what happens when the adversary impersonates a *server*. Here, the simulator must send some c_1 (presumably an encryption of some password pw') *before* the adversary replies with c_2. As before, the simulator can decrypt c_2, recover the password pw in it, and submit this guess to $\mathcal{F}_{\mathsf{pwKE}}$. However, if it turns out that pw is a *correct* guess, the simulator is stuck with a ciphertext c_1 that in all likelihood is an encryption of the wrong password. Not knowing the hashing key hk that $\mathcal{A}$ holds, the simulator cannot predict the value $H_{hk}(c_1, pw)$ that $\mathcal{A}$ will compute (since $(c_1, pw) \notin L$). Thus, the simulator seems to have no way of ensuring that the secret key that $\mathcal{A}$ computes is

[7] This need to extract is not a mere technicality, but is rather quite central to our definition. In particular, this enables us to argue that the level of security achieved is equivalent to the probability of successfully guessing the password, even in the case that related and partially-leaked passwords are used.

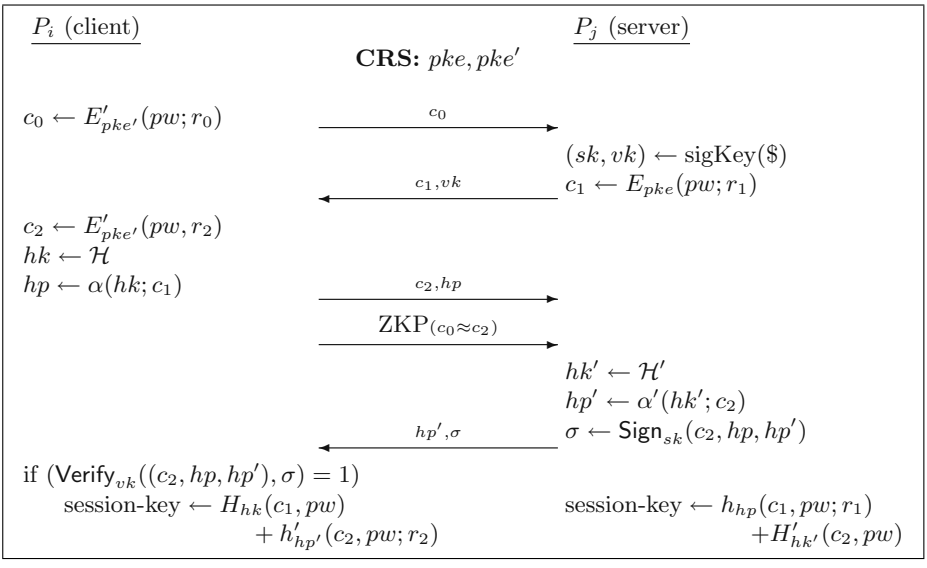

Fig. 4. The core of the universally-composable protocol

the same as the one that the environment gets from the functionality (via the client).[8]

To overcome this problem, we modify the protocol by having the server send a "pre-flow" c_0 which also contains an encryption of the password; i.e., $c_0 = E(pw; r_0)$. Then, when the server later sends c_2, it proves in zero knowledge that c_0 and c_2 are encryptions of the same value. We stress that the session-key is still computed using only c_1 and c_2 and so it is important that consistency hold between these two only (where by consistency, we mean that they are both an encryption of the same password). The modified protocol is sketched in Figure 4. (Note that we switch the "client" and "server" roles so that the client is still the one who sends the first message.)

We now describe the high-level simulation strategy for the modified protocol (and thus why this modification solves the above-described problem):

1. *Case 1 — the adversary $\mathcal{A}$ impersonates the client:* The simulator $\mathcal{S}$ obtains the ciphertext c_0, decrypts it to obtain pw and sends pw to $\mathcal{F}_{\mathsf{pwKE}}$ as the password guess. If this guess is correct, then $\mathcal{S}$ continues the simulation using the same pw and consistency is achieved. Note that the zero-knowledge proof ensures that the ciphertext c_2 later sent by $\mathcal{A}$ is also an encryption of pw (therefore in this case, consistency between c_0 and c_1 implies consistency between c_1 and c_2, as required).

[8] This problem does not arise in the proofs of KOY/GL, since the "simulator" there can just halt and the adversary is declared successful. Our simulator, on the other hand, must continue to simulate both when the adversary fails *and* when it succeeds.

2. *Case 2 — the adversary $\mathcal{A}$ impersonates the server:* In this case, the simulator $\mathcal{S}$ generates the pre-flow ciphertext c_0 as an encryption of some default value (which actually will not be any password). Then, upon receiving c_1 from $\mathcal{A}$, the simulator $\mathcal{S}$ decrypts it to obtain pw and sends pw to $\mathcal{F}_{\mathsf{pwKE}}$ as the password guess. If the guess is correct, then $\mathcal{S}$ generates c_2 to also be an encryption of pw. Notice that c_1 and c_2 are now consistent in that they both encrypt the correct password pw. The only problem remaining in the simulation is that $\mathcal{S}$ is supposed to prove that c_0 and c_2 are indeed consistent (which in this case they are not). It does this by using the *zero-knowledge simulator* for the zero-knowledge proof of consistency. By the zero-knowledge property, this proof is indistinguishable from a real one (and this holds even though the statement in this case is false). We therefore conclude that in the case of a correct password guess, consistency is achieved and in addition, the adversary cannot distinguish its view in the simulation from its view in a real execution.

We stress that Figure 4 is only a sketch of the protocol and does not contain all the details. For example, the full protocol uses *labeled encryption* [29] in order to bind certain protocol information to the ciphertexts (such as the session-id and the verification key of the signature scheme) and in order to prevent other types of man-in-the-middle attacks. (Labels were used implicitly for the same purpose in [24, 18].) A detailed description of the protocol and its proof can be found in the full version, where we also provide a formal statement and proof of the following result:

Theorem 1 (main theorem – informally stated): *Assume the existence of CCA-secure encryption schemes with smooth projective hash functions, and simulation-sound zero-knowledge proofs. Then there exists a protocol in the common reference string model that securely realizes the $\mathcal{F}_{\mathsf{pwKE}}$ functionality in the presence of static-corruption adversaries.*

We remark that all the building blocks of our protocol can be built under the DDH, quadratic residuosity, or N^{th}-residuosity assumptions, so UC-secure password-based key exchange is possible under any of these assumptions. For the most efficient instantiation, we would use encryption and smooth projective hash proofs based on DDH, a collision-resistant hash function for the one-time signature, and a simulation-sound zero-knowledge proof system [17, 25] based on strong RSA. Hence, the end result would rely on all of these assumptions for its security.

Efficiency notes. Considering the protocol as depicted in Figure 4, we emphasize that it suffices to use a (simulation sound) zero-knowledge proof of membership, rather than a proof of knowledge. This allows for efficient solutions; see [17]. Furthermore, it suffices to generate c_0 and c_2 using an encryption scheme that

is only *CPA-secure*, rather than CCA-secure.[9] Thus, the encryption scheme E in Figure 4 (that is used to generate c_1) is CCA-secure, but the encryption scheme E' (that is used to generate c_0 and c_2) is only CPA-secure. Using a CPA-secure scheme for E' provides efficiency improvements in the encryption itself, the projective hashing, and the proof of consistency. In [18] a highly efficient and simple construction of smooth projective hashing was demonstrated for the El Gamal encryption scheme (which is CPA-secure under the DDH assumption). Furthermore, proving consistency of El Gamal encryptions is more efficient than proving consistency of, e.g., CCA-secure Cramer-Shoup encryptions.

4 Additional Results

We describe some additional results that appear in the full version of the paper.

Functionality $\mathcal{F}_{\mathsf{pwSC}}$

The functionality $\mathcal{F}_{\mathsf{pwSC}}$ is parameterized by a security parameter k. It interacts with an adversary S and a set of parties via the following queries.

Upon receiving a $(\mathsf{NewSession}, sid, P_i, P_j, pw, \text{role})$ **query from party** P_i**:**
Send $(\mathsf{NewSession}, sid, P_i, P_j, \text{role})$, to S. In addition, if this is the first NewSession query, or if this is the second NewSession query and there is a record (P_j, P_i, pw'), then record (P_i, P_j, pw) and mark it fresh.

Upon receiving a $(\mathsf{TestPwd}, sid, P_i, pw')$ **query from the adversary** S**:**
If there is a record of the form (P_i, P_j, pw) which is fresh, then do: If $pw = pw'$, then mark the record compromised and reply to S with "correct guess". If $pw \neq pw'$, then mark it interrupted and reply with "wrong guess".

Upon receiving a (Send, sid, m) **query from** P_i**:**
If there is a record of the form (P_i, P_j, pw) then:
- If this record is compromised, or P_i is corrupted, then send (sid, P_i, m) to the adversary.
- If this record is fresh, and there is a record (P_j, P_i, pw') with $pw' = pw$ that is also fresh, then send $(\mathsf{Received}, sid, m)$ to P_j and $(sid, P_i, |m|)$ to the adversary.
- In any other case, send $(sid, P_i, |m|)$ to the adversary.

Upon receiving an $(\mathsf{ExpireSession}, sid)$ **query from** P_i**:**
If there is a record of the form (P_i, P_j, pw), it marks the record as expired.

Fig. 5. The password-based secure channels functionality, $\mathcal{F}_{\mathsf{pwSC}}$

Password-based secure channels. Arguably, the typical use of a key-exchange protocol is the establishment of *secure channels* that enable the parties to communicate privately and reliably. The case of password-based key-exchange is no

[9] In fact, the second encryption in the KOY/GL protocols can be generated using a CPA-secure scheme (e.g., El Gamal) as well; see [23]. This yields the most efficient known password-based key-exchange protocol in the standard model (i.e., without random oracles), albeit under a weaker definition than the one considered here.

exception. In Figure 5, we describe the password-based secure channels functionality $\mathcal{F}_{\mathsf{pwSC}}$. In the full version, we show that our definition of password-based key-exchange suffices for securely realizing this functionality, thus providing additional "justification" for our definition of $\mathcal{F}_{\mathsf{pwKE}}$. The definition of $\mathcal{F}_{\mathsf{pwSC}}$ is analogous to the password-based key-exchange functionality. Notice that if two parties have sent NewSession queries with the same identifiers and passwords, *and* the adversary has not guessed this password or interrupted the session, then the functionality faithfully passes messages from the first party to the second. Furthermore, the adversary learns only the length of the message sent. Thus, the functionality provides *reliable and private communication*, as desired. (The functionality only deals with unidirectional communication from P_i to P_j; it can be repeated in order to obtain bidirectional communication.)

Impossibility of realizing $\mathcal{F}_{\mathsf{pwKE}}$ in the plain model. Our protocol is cast in the common reference string model which assumes some (albeit, rather minimal) trusted setup phase. An important question to ask is whether or not this is *necessary* for obtaining security. In the full version of this paper, we prove that the $\mathcal{F}_{\mathsf{pwKE}}$ functionality cannot be securely realized in the "plain" model (i.e., without using a common reference string or some other augmentation to the basic model). Our proof is similar to the proofs of impossibility in [10]. The basic idea is as follows. Consider an environment that internally runs the code of one of the honest parties. The ideal-model simulator for such an environment must succeed while interacting with it in the same way that real parties interact. (This holds by the definition of the UC framework which requires a black-box simulator which cannot rewind.) Now, if simulation can be carried out in such a scenario, then it can also be carried out while interacting with a real honest party (because the specific environment we have chosen behaves like an honest party). This means that anything the ideal-model simulator/adversary can do with respect to the environment, a real-model adversary can do with respect to an honest party. In particular, in order for the simulation to succeed, the ideal-model simulator must be able to set the output session-key to be the same as the key output by the ideal functionality. Thus, a real-model adversary can also do this, in contradiction to the security requirements of the key-exchange protocol.

References

1. B. Barak, Y. Lindell, and T. Rabin, Protocol Initialization for the Framework of Universal Composability. Manuscript. Available from the ePrint archive, report 2004/006 from `http://eprint.iacr.org`.
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. *Advances in Cryptology – Crypto 1998*, LNCS vol. 1462, Springer-Verlag, pp. 26–45, 1998
3. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. *Advances in Cryptology – Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 139–155, 2000.
4. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *Advances in Cryptology – Crypto 1993*, LNCS vol. 773, Springer-Verlag, pp. 232–249, 1993.

5. S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. Proc. *IEEE Security and Privacy*, IEEE, pp. 72–84, 1992.
6. V. Boyko. On All-or-Nothing Transforms and Password-Authenticated Key Exchange. PhD thesis, MIT, EECS department, 2000.
7. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password Authentication and Key Exchange Using Diffie-Hellman. *Advances in Cryptology – Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 156–171, 2000.
8. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 136–145, 2001.
9. R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. *Advances in Cryptology – Eurocrypt 2002*, LNCS vol. 2332, Springer-Verlag, pp. 337–351, 2002.
10. R. Canetti, E. Kushilevitz and Y. Lindell. On the Limitations of Universal Composable Two-Party Computation Without Set-Up Assumptions. In *EUROCRYPT 2003,* Springer-Verlag (LNCS 2656), pages 68–86, 2003.
11. R. Canetti and T. Rabin. Universal Composition with Joint State. *Advances in Cryptology – Crypto 2003*, LNCS vol. 2729, Springer-Verlag, pp. 265–281, 2003.
12. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. *Advances in Cryptology – Crypto 1994*, LNCS vol. 839, Springer-Verlag, pp. 174–187, 1994.
13. R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. *Advances in Cryptology – Eurocrypt 2002*, LNCS vol. 2332, Springer-Verlag, pp. 45–64, 2002.
14. A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust Non-Interactive Zero-Knowledge. *Advances in Cryptology – Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 566–598, 2001.
15. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM J. Computing* 30(2): 391–437, 2000.
16. S. Even, O. Goldreich, and S. Micali. On-Line/Off-Line Digital Signatures. *J. Cryptology* 9(1):35-67, 1996.
17. J. Garay, P. MacKenzie, and K. Yang. Strengthening Zero-Knowledge Protocols Using Signatures. *Advances in Cryptology – Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 177–194, 2003.
18. R. Gennaro and Y. Lindell. A Framework for Password-Based Authenticated Key Exchange. *Advances in Cryptology – Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 524–543, 2003.
19. O. Goldreich and Y. Lindell. Session-Key Generation using Human Passwords Only. *Advances in Cryptology – Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 408–432, 2001.
20. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences* 28:270–299, 1984.
21. L. Gong, M. Lomas, R. Needham, and J. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5): 648–656, 1993.
22. Shai Halevi and Hugo Krawczyk. Public-Key Cryptography and Password Protocols. *ACM Trans. on Information and Systems Security,* 2(3):230–268, 1999.
23. J. Katz, P. MacKenzie, G. Taban, and V. Gligor. Two-Server Password-Only Authenticated Key Exchange. Manuscript, Jan. 2005.

24. J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. *Advances in Cryptology – Eurocrypt 2001*, LNCS vol. 2045, Springer-Verlag, pp. 475–494, 2001.
25. P. MacKenzie and K. Yang. On Simulation-Sound Trapdoor Commitments. *Advances in Cryptology – Eurocrypt 2004*, LNCS vol. 3027, Springer-Verlag, pp. 382–400, 2004. Available from the ePrint archive, report 2003/252 from `http://eprint.iacr.org`.
26. M.H. Nguyen and S. Vadhan. Simpler Session-Key Generation from Short Random Passwords. Proceedings of the *1st Theory of Cryptography Conference* (TCC'04), LNCS vol. 2951, Springer-Verlag, pp. 442–445, 2004.
27. C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. *Advances in Cryptology – Crypto 1991*, LNCS vol. 576, Springer-Verlag, pp. 433–444, 1991.
28. A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. *40th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 543–553, 1999.
29. V. Shoup. A Proposal for an ISO Standard for Public Key Encryption (version 2.1). December, 2001. Available from the ePrint archive, report 2001/112 from `http://eprint.iacr.org`.

Mercurial Commitments with Applications to Zero-Knowledge Sets

Extended Abstract

Melissa Chase[1], Alexander Healy[2], Anna Lysyanskaya[1], Tal Malkin[3], and Leonid Reyzin[4]

[1] Brown University
{mchase,anna}@cs.brown.edu
[2] Harvard University
ahealy@fas.harvard.edu
[3] Columbia University
tal@cs.columbia.edu
[4] Boston University
reyzin@cs.bu.edu

Abstract. We introduce a new flavor of commitment schemes, which we call *mercurial commitments*. Informally, mercurial commitments are standard commitments that have been extended to allow for *soft* decommitment. Soft decommitments, on the one hand, are not binding but, on the other hand, cannot be in conflict with true decommitments.

We then demonstrate that a particular instantiation of mercurial commitments has been implicitly used by Micali, Rabin and Kilian to construct *zero-knowledge sets*. (A *zero-knowledge set* scheme allows a Prover to (1) commit to a set S in a way that reveals nothing about S and (2) prove to a Verifier, in zero-knowledge, statements of the form $x \in S$ and $x \notin S$.) The rather complicated construction of Micali et al. becomes easy to understand when viewed as a more general construction with mercurial commitments as an underlying building block.

By providing mercurial commitments based on various assumptions, we obtain several different new zero-knowledge set constructions.

1 Introduction

1.1 Mercurial Commitments

A traditional cryptographic commitment is often compared to a safe. The sender places a particular value in the safe, locks it and gives it to the recipient. The recipient cannot see the value, but is assured that it will not change while inside the safe. Then, whenever the sender chooses to, he can reveal the secret code needed to open the safe, enabling the recipient to retrieve the hidden value. The two usual properties of commitments are therefore *hiding* and *binding*: the sender is bound to the message, but the message is hidden from the recipient.

We propose a variant of traditional commitments, where the opening protocol is two-tiered. Partial opening, which we call "teasing", is not truly binding: it

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 422–439, 2005.

is possible for the sender to come up with a commitment that can be teased to any value of the sender's choice. True opening, on the other hand, is binding in the traditional (computational) sense: it is infeasible for the sender to come up with a commitment that he can open to two different values.

Despite the fact that a commitment can potentially be teased to any value, a tease is not merely a meaningless assertion. A tease of a commitment to a value m is a guarantee that the commitment cannot be opened to any value other than m. In other words, the recipient of a tease knows that if the commitment can be opened at all, then it will be to the same value. It is infeasible for the sender to come up with a commitment that can be teased to m_1 and opened to $m_2 \neq m_1$.

This immediately implies, of course, that if the sender can open a commitment at all, then it can be teased to only one value. Thus, the sender must choose, at the time of commitment, whether to "soft-commit," so as to be able to tease to multiple values but not open at all, or to "hard-commit," so as to be able to tease and to open to only one particular value. The recipient, however, cannot tell which of the two options the sender has chosen (this is ensured by the hiding property).

We call this new primitive *mercurial commitment*.

Mercurial commitments are different from trapdoor or chameleon commitments of [BCC88]. All chameleon commitments are equivocal whenever the sender knows a trapdoor for the commitment scheme. In mercurial commitments, on the other hand, the sender is given the choice, at the time of commitment, whether to make the commitment equivocal or binding. Furthermore, in chameleon commitments, equivocated and regular decommitments look the same to the recipient; whereas in mercurial commitments, the recipient may be content with the decommitment that may have been equivocated (tease), or may require the stronger full decommitment (open).

Note that mercurial commitments directly imply conventional commitments as a special case, when only hard-commit and open are used (and the soft-commit and tease functionalities are ignored).

We have not yet addressed the hiding property of mercurial commitments. For our application, we need a very strong hiding property, namely, simulatability (which we can provide in the shared-random-string or trusted-parameters model[1], or else interactively). However, such strong hiding does not seem to be an essential property of mercurial commitments, and it is conceivable that, if mercurial commitments find other applications, weaker hiding properties will suffice.

We formally define mercurial commitments in Section 2.1. We provide four constructions in Section 2.2: based on general (possibly noninteractive) zero-

[1] The shared-random-string model assumes that a uniform random string is available for all parties to use. The trusted-parameters model assumes that a public string from some (possibly complex) distribution has been produced and is available for all parties to use; furthermore the (uniform) coins used to produce that string are unknown to the parties (for instance, such a string could be a product n of two large primes p and q, where the primes themselves are unknown to the parties).

knowledge, claw-free permutations, discrete logarithms, and factoring respectively. The last two constructions are efficient enough to be useable in practice.

We distilled the notion of mercurial commitments out of the zero-knowledge set construction of [MRK03], where a particular construction (namely, the one based on discrete logarithms) of mercurial commitments is implicitly used. We believe that abstracting this notion and separating its construction from the construction of zero-knowledge sets themselves is beneficial. First, as we demonstrate in Section 3.2, the [MRK03] construction of zero-knowledge sets becomes conceptually simpler when mercurial commitments are used as a building block. Second, when mercurial commitments can be studied in isolation, it is much easier to come up with novel constructions for them, and therefore also for zero-knowledge sets. Finally, mercurial commitments are interesting independently of this specific application because of their potentially broader applicability.

1.2 Zero-Knowledge Sets

Zero-knowledge sets (ZKS) were recently introduced by Micali, Rabin, and Kilian [MRK03]. ZKS allow a prover to commit to an arbitrary finite set S in such a way that for any string x he can provide an efficient sound proof of whether $x \in S$ or $x \notin S$, without revealing any knowledge beyond this membership assertion. That is, the recipient (verifier) of the proof learns nothing else about the set S, not even the size of S. We elaborate on the formal definition of ZKS in Section 3.1.

As pointed out by [MRK03], the notion of zero-knowledge sets can be extended to zero-knowledge elementary databases, where each element $x \in S$ has a value $v(x)$ associated with it. After committing to S, the prover can provide an efficient proof for each x of either "$x \in S$ and $v(x) = v$", or "$x \notin S$", without revealing any further information. Sets, of course, are a special case of this, where the value associated with each $x \in S$ is 1. Throughout this paper, we use ZKS to refer also to the more general zero-knowledge elementary databases.

Micali, Rabin, and Kilian give a construction of zero-knowledge sets under the discrete logarithm assumption in the shared random string model. This construction is noninteractive (i.e., both the initial commitment and query answers require a single message from the prover to the verifier) with $O(k^2)$-bit proofs for security parameter k. They do not show how to remove the number-theoretic details of their construction, and leave open whether constructions not based on the discrete logarithm assumption are possible at all.

It is an interesting problem to consider what alternative constructions are possible, and under what assumptions these constructions can be realized.

Ostrovsky, Rackoff and Smith [ORS04] provide constructions for consistent database queries, which allow the prover to commit to a database, and then provide query answers that are provably consistent with the commitment. They also consider the problem of adding privacy to such protocols. Their constructions can handle queries much more general than just membership queries; they yield two constructions of ZKS as special cases. The first construction is a feasibility result, showing that interactive ZKS can be built out of (public) collision-resistant hash

functions (CRHF) and zero-knowledge proofs of NP statements (which require only one-way functions, which are implied by CRHF); noninteractive ZKS can be built in the shared random string model out of CRHF and noninteractive zero-knowledge. The second construction is more efficient, based on the assumptions of CRHF and homomorphic commitments. Unfortunately, it requires interaction (which can be removed in the random oracle model) and requires the prover to keep a counter t of the number of queries asked so far. (For security parameter k, the proofs are of size $O(tk^4)$ and, in particular, grow with t.[2])

We provide an alternative proof of the same feasibility result, as well as more efficient constructions based on different assumptions, as detailed next.

1.3 Zero-Knowledge Sets from Mercurial Commitments

We describe the work of [MRK03] in light of our new primitive, thus showing how to construct zero-knowledge sets based on mercurial commitments and collision-resistant hash functions. Different instantiations of mercurial commitments will result in different ZKS constructions with different security assumptions and efficiency.

Instantiating our ZKS construction with mercurial commitments based on general zero-knowledge gives an alternative proof of the feasibility of ZKS from general assumptions (as mentioned above, another such proof was given independently by [ORS04]). It shows that (noninteractive) ZKS can be constructed in the shared random string model by using as building blocks noninteractive zero-knowledge (NIZK) proofs [BDMP91, FLS99], (conventional) commitment schemes (which are anyway implied by NIZK), and CRHF.[3] If one is willing to add interaction to the revealing (membership proof) phase of ZKS, our construction shows that CRHF and interactive ZKS are equivalent (because NIZK can be replaced with regular zero-knowledge proofs, which can be based on one-way functions, which are implied by CRHF; on the other hand, it is quite clear that CRHF is necessary for ZKS, because the initial commitment to the set must be collision-resistant). Unfortunately, the above discussion applies merely to feasibility results; none of these constructions is practical.

Instantiating our ZKS construction with mercurial commitments based on claw-free permutations gives ZKS in the trusted parameters model with proof length $O(k^3)$. The construction based on factoring further improves the efficiency, giving ZKS with proof length $O(k^2)$ and verification time $O(k^4)$, suitable for practical implementation in the trusted parameters model.

---

[2] The proof size given in [ORS04] is $O(tdsk^2)$, where s is a bound on the length of each key x, and d is a bound on logarithm of the set size. However, in order to hide the set size, we must first hash each key to a k-bit value, and set $d = k$.

[3] It is known how to construct NIZK proofs based on the existence of trapdoor permutations (TDP) [FLS99, BY96], or based on the existence of verifiable random functions (VRF) [GO92, MRV99]. TDP and VRF are, as far as we currently know, incomparable assumptions. Indeed, VRFs can be constructed based on gap-Diffie-Hellman groups [Lys02], while no trapdoor permutation is known based on such groups.

For the case of ZKS from discrete-logarithm-based mercurial commitments (which are the ones implicitly used in [MRK03]), we provide a constant-factor improvement over the [MRK03] construction by utilizing a hash function better suited for such commitments. The resulting construction is within the realm of practical implementation in the shared random string model, requiring proofs of length $O(k^2)$ and verification time $O(k^4)$ (constants hidden by O here are fairly small and are further analyzed in Section 5).

2 The New Primitive: Mercurial Commitments

2.1 Definition

As we describe in the introduction, a mercurial commitment is a commitment scheme with additional features. The first feature is that, in addition to the usual algorithm for opening a commitment, there is also an algorithm to partially open, or *tease*. The partial decommitment of a commitment C to a value x means, in essence, that if C can be opened at all, then it can be opened only to x. The second feature of a mercurial commitment scheme is that a commitment C can be formed in two ways: it may be a *hard* commitment, that is, a commitment that can be opened (and teased) in only one way; or a *soft* commitment that cannot be opened at all, but can be teased to any value. Let us now describe this more formally.

A mercurial commitment scheme consists of the following algorithms:

SETUP This is a randomized algorithm run by a trusted third party that sets up the public key for the commitment. We write $PK \leftarrow \text{SETUP}(1^k)$. The chosen public key PK defines the (efficiently samplable) domain of possible committed values. Let us denote this domain D_{PK}. If this algorithm merely outputs its random coins, then the mercurial commitment scheme is in the *shared random string* model. Else it is in the stronger *trusted parameters* model.

HARD-COMM This is the *deterministic* algorithm used to commit to a value. It takes as input the public key PK, a value $x \in D_{PK}$, and a random string r, and outputs the commitment C. We write $C = \text{HARD-COMM}(PK, x, r)$.

SOFT-COMM This is the *deterministic* algorithm used to soft-commit. That is to say, a value produced by this algorithm is not really a commitment because it can never be opened. But it can be partially opened (teased) to any value of the committer's choice. This algorithm takes as input the public key PK and the random string r, and outputs a value C. We write $C = \text{SOFT-COMM}(PK, r)$.

TEASE This is the randomized algorithm for partially opening (teasing) a hard or soft commitment. On input (PK, x, r, C), where C is either a hard commitment to x with string r, or a soft commitment with string r, TEASE outputs the partial decommitment τ for teaser value x. We write $\tau \leftarrow \text{TEASE}(PK, x, r, C)$.

VER-TEASE This is the algorithm that either accepts or rejects the partial decommitment τ to teaser value x. It takes as input the public key PK, the commitment C, and the values x and τ.

OPEN This algorithm opens the commitment C. If $C = \text{HARD-COMM}(PK, x, r)$, then on input (PK, x, r, C), OPEN will output the decommitment π for the committed value x. We write $\pi \leftarrow \text{OPEN}(PK, x, r, C)$.

VER-OPEN This is the algorithm that either accepts or rejects the decommitment π to the value x. It takes as input the public key PK, the commitment C, and the values x and π.

As usual for commitment schemes, we require three properties: (1) correctness: VER-TEASE will always accept the correctly formed partial decommitment τ of C for the correct teaser value x, and VER-OPEN will always accept the correctly formed decommitment π of C for the correct x; (2) binding: no adversary can create C such that it can be opened to two different values, and no adversary can create C such that it can be opened to one value but partially decommitted (teased) to another value; (3) hiding: no adversary can learn whether C is a soft commitment or hard commitment, and in case it is a hard commitment, no adversary can learn the committed value x; moreover, we require that there be a *simulator* that will be able to form C in such a way that it can not only partially decommit (tease) it to any teaser value, but also open it to any value, such that the view of the receiver will be the same whether it is talking to the committer or to the simulator.

More precisely:

Definition 1. *A set of algorithms* SETUP, HARD-COMM, SOFT-COMM, TEASE, OPEN, VER-TEASE *and* VER-OPEN *satisfies the* correctness *property of mercurial commitments if for all* $PK \in \text{SETUP}(1^k)$

- *Correctness for* HARD-COMM*: For all* $x \in D_{PK}$*, for all strings* r*, if* $C = \text{HARD-COMM}(PK, x, r)$*, then*
  - *for all* $\tau \in \text{TEASE}(PK, x, r, C)$, $\text{VER-TEASE}(PK, C, x, \tau) = OK$*;*
  - *for all* $\pi \in \text{OPEN}(PK, x, r)$, $\text{VER-OPEN}(PK, C, x, \pi) = OK$*;*
- *Correctness for* SOFT-COMM*: For all* r*, if* $C = \text{SOFT-COMM}(PK, r)$*, then for all* $x \in D_{PK}$*, for all* $\tau \in \text{TEASE}(PK, x, r, C)$, $\text{VER-TEASE}(PK, C, x, \tau) = OK$.

Definition 2. *A set of algorithms* SETUP, VER-TEASE *and* VER-OPEN *satisfies the* binding *property of mercurial commitments if for all probabilistic polynomial-time nonuniform adversaries* $\{A_k\}$ *there exists a negligible function* ν *such that*

$$\Pr[PK \leftarrow \text{SETUP}(1^k); (C, x, x', \pi, \tau) \leftarrow A_k(PK) \ : \\ \text{VER-OPEN}(PK, C, x, \pi) = OK \wedge \\ (\text{VER-OPEN}(PK, C, x', \tau) = OK \vee \text{VER-TEASE}(PK, C, x', \tau) = OK) \wedge \\ x \neq x' \in D_{PK}] = \nu(k)$$

Definition 3. *A set of algorithms* Setup, Hard-Comm, Soft-Comm, Tease, Open *satisfies the* hiding (simulatability) *property of mercurial commitment if*

1. *There exists a set of algorithms* Sim-Setup, Sim-Commit, Sim-Tease, Sim-Open *with the following specifications:*

   Sim-Setup *This is a randomized algorithm that, in addition to creating the commitment public key PK, also outputs a trapdoor key TK that allows the simulator some extra power that the legitimate committer does not have. We write* $(PK, TK) \leftarrow$ Sim-Setup(1^k).

   Sim-Commit *This is the deterministic algorithm that the simulator uses to compute a commitment. Besides (PK, TK), it takes a random string r as input. We write* $C =$ Sim-Commit(PK, TK, r).

   Sim-Tease *This is the algorithm that the simulator uses to compute a partial decommitment for any value $x \in D_{PK}$. On input (PK, TK, r, x), it gives the partial decommitment τ for the commitment* $C =$ Sim-Commit (PK, TK, r). *We write* $\tau \leftarrow$ Sim-Tease(PK, TK, r, x).

   Sim-Open *This is the algorithm that the simulator uses to compute a decommitment for any value $x \in D_{PK}$. On input (PK, TK, r, x), it outputs the decommitment π for the commitment* $C =$ Sim-Commit(PK, TK, r). *We write* $\pi \leftarrow$ Sim-Open(PK, TK, r, x).

2. *Let the following algorithms be defined as follows:*

   Committer$_{PK}$ *The committer algorithm C_{PK} is a stateful algorithm that responds to requests to hard- and soft-commit to specific values by running* Hard-Comm *and* Soft-Comm, *and then, on request runs the* Tease *and* Open *algorithms on the corresponding commitments. It also maintains a list L of commitments issued so far. Initially, list L is empty. Here is how C_{PK} responds to various inputs:*

   - *On input* (Hard-Comm$, x$)*, choose a random string r. Compute* $C =$ Hard-Comm(PK, x, r). *Store* (Hard-Comm$, C, x, r$) *in the list L. Output C.*
   - *On input* (Soft-Comm)*, choose a random string r. Compute* $C =$ Soft-Comm(PK, r). *Store* (Soft-Comm$, C, r$) *in the list L. Output C.*
   - *On input* (Tease$, C, x'$)*:*
     - *Check if $C \in L$. If it is not, output "fail." Else, retrieve the record corresponding to C.*
     - *If C's entry on the list is of the form* (Hard-Comm$, C, x, r$)*: if $x \neq x'$, output "fail." Otherwise, output* $\tau =$ Tease(PK, x, r, C).
     - *Else if C's entry on the list is of the form* (Soft-Comm$, C, r$)*: output* $\tau =$ Tease(PK, x', r, C).
   - *On input* (Open$, C, x$)*, check if for some r,* (Hard-Comm$, C, x, r$) *is on the list. If it is not, output "fail." Else, output* Open(PK, x, r).

   Simulator$_{(PK,TK)}$ *The simulator $S_{(PK,TK)}$ answers the same types of queries as the Committer C_{PK}, but by running different algorithms. It also maintains the* same *list L, initialized to empty.*

- *On input* (HARD-COMM, x)*, choose a random string* r*. Compute* $C =$ SIM-COMMIT(PK, TK, r)*. Store* (HARD-COMM, C, x, r) *in the list* L*. Output* C*.*
- *On input* (SOFT-COMM)*, choose a random string* r*. Compute* $C =$ SIM-COMMIT(PK, TK, r)*. Store* (SOFT-COMM, C, r) *in the list* L*. Output* C*.*
- *On input* (TEASE, C, x')*:*
  - *Check if* $C \in L$*. If it is not, output "fail." Else, retrieve the record corresponding to* C*.*
  - *If* C*'s entry on the list is of the form* (HARD-COMM, C, x, r)*: if* $x \neq x'$*, output "fail." Otherwise, output* $\tau \leftarrow$ SIM-TEASE(PK, TK, x, r, C)*.*
  - *Else if* C*'s entry on the list is of the form* (SOFT-COMM, C, r)*: output* $\tau \leftarrow$ SIM-TEASE(PK, TK, x', r, C)*.*
- *On input* (OPEN, C, x)*, check if for some* r*,* (HARD-COMM, C, x, r) *is on the list. If it is not, output "fail." Otherwise, output* SIM-OPEN (PK, TK, x, r)*.*

Then no polynomial-time distinguisher can tell whether he is talking to a Committer or to a Simulator. Namely, for all probabilistic polynomial-time families of oracle Turing machines $\{D_k^?\}$*, there exists a negligible function* $\nu(k)$ *such that*

$$\Pr[PK_0 \leftarrow \text{SETUP}(1^k); (PK_1, TK) \leftarrow \text{SIM-SETUP}(1^k); O_0 = C_{PK_0}; O_1 = S_{(PK_1, TK)}; b \leftarrow \{0,1\}; b' \leftarrow D_k^{O_b}(pk_b) \ : \ b = b'] = 1/2 + \nu(k)$$

(In this definition, we create two oracles: O_0 *is a Committer, and* O_1 *is a Simulator. Then the distinguisher interacts with a randomly chosen oracle, and has to guess which oracle it is talking to.)*

Remarks. Note that the notion of simulatability can be defined in three flavors: perfect, statistical, and computational, corresponding to the strength of the distinguisher D. Above, we gave the definition for the computational flavor since it is the least restrictive. Also note that the definition above is noninteractive. The definition can be extended to an interactive setting, where the decommitment (opening or teasing) is interactive. Throughout the paper, in order to keep the presentation clean, we continue by default to consider noninteractive mercurial commitments (and noninteractive ZKS), and only mention the interactive case in side remarks.

Definition 4 (Mercurial commitment scheme).
Algorithms SETUP, HARD-COMM, SOFT-COMM, TEASE, OPEN, VER-TEASE *and* VER-OPEN *constitute a mercurial commitment scheme if they satisfy the correctness, binding, and hiding (simulatability) properties.*

2.2 Constructions

From General Assumptions. We construct mercurial commitments based on a many-theorem noninteractive zero-knowledge proof system [BDMP91, FLS99]. Such a proof system can be constructed from any trapdoor permutation (TDP) [BDMP91, BY96], or from a verifiable random function (VRF) [GO92, MRV99]. Existence of TDPs and existence of VRFs are, as far as we know, incomparable assumptions, since VRFs can be constructed based on gap-Diffie-Hellman groups [Lys02], while no trapdoor permutation is known based on such groups. This construction is in the shared random string model (not in the trusted parameters model).

Suppose that we are given a many-theorem noninteractive zero-knowledge (NIZK) proof system for an NP-complete language L. This proof system operates in the public random string model, and consists of polynomial-time algorithms Prove and Verify. Further suppose that we are given a conventional noninteractive unconditionally binding commitment scheme, consisting of algorithms (Comm-Setup, Commit). Note that such a commitment scheme is already implied by the existence of NIZK, because NIZK implies OWFs, and in the public-random-string model, OWFs imply setup-free unconditionally binding bit commitment [Nao91, HILL99]. More detailed definitions of these standard building blocks, NIZK and commitment schemes, are given in the full version of the paper.

We now describe a (noninteractive) mercurial commitment scheme based on a NIZK proof system and any noninteractive commitment scheme. The idea of this construction is simple: a mercurial commitment will consist of two conventional commitments. The first one determines whether it is a hard-commit or soft-commit. The second one determines the value itself in case of hard-commit. To tease, simply prove (using NIZK) that "either this is a soft-commit, or the committed value is x." To open, prove (using NIZK) that "this is a hard-commit to x." Correctness will follow from the correctness properties of the NIZK and of the commitment scheme. The binding property will follow from the fact that the commitment scheme is unconditionally binding, and from the soundness of the NIZK. Simulatability will follow from the security of the commitment scheme and from the zero-knowledge property (i.e., existence of the zero-knowledge simulator) of the NIZK.

An (easily produced) more formal description of the above scheme is contained in the full version of the paper. We thus obtain the following theorem.

Theorem 1. *The construction above is a mercurial commitment scheme, assuming the underlying primitives satisfy the definitions of NIZK proofs for NP and unconditionally binding commitment schemes.*

As noted in the introduction, the same construction can be used to achieve interactive mercurial commitments, from standard commitments and (interactive) zero knowledge proofs. Since both these building blocks are implied by OWF, the construction yields interactive mercurial commitments based on OWF.

From Claw-Free Trapdoor Bijections. We now construct a mercurial bit-commitment scheme under the assumption that there exist *claw-free trapdoor bijections*.[4] Specifically, slightly generalizing the notion of claw-free permutations of [GMR88], we assume that there exist indexed families of bijections $\{f_i\}_{i\in I\subseteq\{0,1\}^n}$, $f_i : D_{f_i} \to R_i$ and $\{g_i\}_{i\in I\subseteq\{0,1\}^n}$, $g_i : D_{g_i} \to R_i$, and an efficiently computable distribution Δ on pairs $(i, t_i) \in \{0,1\}^n \times \{0,1\}^{\mathrm{poly}(n)}$ such that:

- t_i is trapdoor information that allows f_i to be inverted efficiently.
- f_i and g_i are claw-free. That is, when given i sampled according to Δ, no efficient algorithm can, with nonnegligible probability, find $s \in D_{f_i}$ and $s' \in D_{g_i}$ such that $f_i(s) = g_i(s')$.

Employing this assumption, we construct mercurial bit-commitmens:

- $PK = \textsc{Setup}(1^n) = i$ where (i, t_i) is sampled from Δ
- Using randomness $(r_0, r_1) \in D_{f_i} \times D_{g_i}$, $\textsc{Hard-Comm}(i, 0, (r_0, r_1)) = (f_i(r_0), g_i(r_1))$ and $\textsc{Hard-Comm}(i, 1, (r_0, r_1)) = (g_i(r_1), f_i(r_0))$
- Using randomness $(r_0, r_1) \in D_{f_i} \times D_{f_i}$, $\textsc{Soft-Comm}(i, (r_0, r_1)) = (f_i(r_0), f_i(r_1))$
- For hard commitment $C = (C_0, C_1)$, $\tau = \textsc{Tease}(i, x, (r_0, r_1), (C_0, C_1)) = r_0$.
- For soft commitment $C = (C_0, C_1)$, $\tau = \textsc{Tease}(i, x, (r_0, r_1), (C_0, C_1)) = r_x$.
- $\textsc{Ver-Tease}(i, x, \tau, (C_0, C_1))$ checks that $C_x = f_i(\tau)$.
- To open a hard commitment $C = (C_0, C_1)$ to x, created using the random string (r_0, r_1), $\pi = \textsc{Open}(i, x, (r_0, r_1), (C_0, C_1)) = (x, r_0, r_1)$.
- Given a decommitment $\pi = (x, r_0, r_1)$, $\textsc{Ver-Open}(i, x, \pi, (C_0, C_1))$ checks $C_x = f_i(r_0)$ and $C_{1-x} = g_i(r_1)$.

The correctness of this commitment scheme is immediate from the above descriptions. Furthermore, it is clear that these commitments are hiding since all commitments are pairs of completely random elements of R_i. That hard commitments are binding follows from the assumption that f_i and g_i are claw-free.

It remains to show that this commitment scheme is simulatable. The key step in showing simulatability is to note that if t_i (i.e. the trapdoor for f_i) is known, then one can easily compute $f_i^{-1}(s)$ for any given $s \in R_i$, and in particular, one can produce an r' such that $s = f_i(r')$, even if s was chosen to be $g_i(r)$ for some random $r \leftarrow D_{g_i}$. The details are provided in the full version of the paper.

Claw-free trapdoor bijections are an established cryptographic primitive [GMR88]. They are commonly realized under the assumption that factoring is hard. However, under the factoring assumption one can construct much more efficient mercurial commitments, as we do later in this section. Nonetheless, the above construction based on a claw-free pair is valuable because the existence of a claw-free pair may be viewed as a generic assumption independent of the difficulty of factoring. Indeed, the assumption seems reasonable generically: the

[4] Note that in contrast to the construction of the previous section, here we construct a bit-commitment scheme, i.e. we commit only to values $x \in \{0, 1\}$.

less the functions f_i and g_i have to do with each other, the more plausible the assumption. Note that only f_i requires a trapdoor, so g_i may be some, completely unrelated, one-way bijection. It may be reasonable to assume that it is infeasible to find a claw in such a case.

From the Discrete Logarithm Assumption. The following mercurial commitment scheme relies on the intractability of the discrete logarithm problem in a group G of prime order. When G is taken to be the subgroup of size q of $\mathbb{Z}_p^*$ where $q|(p-1)$ (i.e., G is the group of d-th order residues in $\mathbb{Z}_p^*$ for a prime $p = dq+1$), this mercurial commitment scheme is implicit in the Zero-Knowledge Sets construction of [MRK03]. Indeed, combining this mercurial commitment with the Zero-Knowledge Set construction of the next section yields essentially the same construction as [MRK03].

Recall that the Pedersen commitment scheme [Ped92] employs two randomly chosen generators, $g, h \leftarrow G$, and a commitment to a message $m \in \{0, 1, \ldots, |G|-1\}$ is computed by selecting a random $r \leftarrow \{0, 1, \ldots, |G|-1\}$ and letting the commitment be $g^m h^r$. The commitment is opened by revealing the message m and the random exponent r. It is not hard to show that if the committer can open this commitment in more than one way, then he can easily compute $\log_g(h)$, a task which is presumed to be intractable. On the other hand, if the committer knows $\log_g(h)$, then he can easily open a supposed commitment $c = g^k \in G$ to any message m by producing the pair $(m, (k-m)/\log_g(h) \bmod |G|)$. This observation is essential to the following mercurial commitment scheme which is based on the Pedersen commitment.

- $\textsc{Setup}(1^k)$ selects G and $(g, h) \leftarrow G \times G$.
- The hard commitment (with random string $(r_0, r_1) \leftarrow \{0, 1, \ldots, |G|-1\}^2$) is simply the Pedersen commitment using the generator pair (g, h^{r_1}): $\textsc{Hard-Comm}((g, h), x, (r_0, r_1)) = (g^x (h^{r_1})^{r_0}, h^{r_1})$
- The soft commitment (with $(r_0, r_1) \leftarrow \{0, 1, \ldots, |G|-1\}^2$) is $\textsc{Soft-Comm}((g, h), (r_0, r_1)) = (g^{r_0}, g^{r_1})$.
- If $C = (C_0, C_1)$ is hard, then $\tau = \textsc{Tease}((g, h), x, (r_0, r_1), (C_0, C_1)) = r_0$.
- If $C = (C_0, C_1)$ is soft, then $\tau = \textsc{Tease}((g, h), x, (r_1, r_2), (C_0, C_1)) = (r_0 - x)/r_1 \bmod |G|$.
- In either case, $\textsc{Ver-Tease}((g, h), (C_0, C_1), x, \tau)$ checks that $C_0 = g^x \cdot C_1^{\tau}$.
- $\textsc{Open}$ computes (π_0, π_1) as $\textsc{Open}((g, h), x, (r_0, r_1), (C_0, C_1)) = (r_0, r_1)$.
- Finally, verification is similar to Pedersen's, with an additional step to ensure that the second generator C_1 was chosen as a known power of h, and hence that $\log_g(C_1)$ is not known to the committer: $\textsc{Ver-Open}((g, h), (C_0, C_1), x, (\pi_0, \pi_1))$ checks that $C_0 = g^x \cdot C_1^{\pi_0}$ and that $C_1 = h^{\pi_1}$.

The correctness of the above algorithms is easily verified. The proof that hard commitments are binding is just as with the Pedersen commitment; indeed, the ability to open a commitment $C = (C_0, C_1)$ in two ways implies knowledge of $\log_g(h)$. This scheme is clearly hiding because all commitments consist of random elements from $G \times G$. As for simulatability, the simulator simply needs to set up g, h and TK as $g \leftarrow G$, $TK \leftarrow \{0, 1, \ldots, |G|-1\}$ and $h = g^{TK}$. A more detailed description is contained in the full version of the paper.

From the Hardness of Factoring. Our final construction is based on the hardness of factoring. Like the discrete logarithm construction, this scheme commits to many bits simultaneously. This is a modification of the trapdoor commitment construction implicit in the GMR signature scheme [GMR88]. We note that a similar mercurial commitment scheme (based on RSA rather than factoring, but allowing for interesting extensions based on the Strong RSA assumption) was independently discovered by Gennaro and Micali [GM05].

The mercurial commitment scheme runs as follows:

- Let the message space be $\{0,1\}^\ell$.
- $\textsc{Setup}(1^n)$ chooses an RSA modulus $N = pq$, where $p \equiv q \equiv 3 \pmod 4$, and a random element $y \in \mathbb{Z}_N^*$. Let $U = y^2$. $PK = (N, U)$.
- Using randomness $(r_0, r_1) \in \mathbb{Z}_N^* \times \mathbb{Z}_N^*$, $\textsc{Hard-Comm}((N,U), m, (r_0, r_1)) = (Ur_0^2, r_1^{2^\ell}(Ur_0^2)^m)$.
- Using randomness $(r_0, r_1) \in \mathbb{Z}_N^* \times \mathbb{Z}_N^*$, $\textsc{Soft-Comm}((N,U), (r_0, r_1)) = (r_0^{2^\ell}, r_1^{2^\ell})$.
- If $C = (C_0, C_1)$ is hard, then $\tau = \textsc{Tease}((N,U), m, (r_0, r_1), (C_0, C_1)) = r_1$.
- If $C = (C_0, C_1)$ is soft, then $\tau = \textsc{Tease}((N,U), m, (r_0, r_1), (C_0, C_1)) = r_1 r_0^{-m}$.
- $\textsc{Ver-Tease}((N,U), m, \tau, (C_0, C_1))$ checks that $C_1 = C_0^m(\tau)^{2^\ell}$.
- $\textsc{Open}$ computes π as $\pi = \textsc{Open}((N,U), m, (r_0, r_1), (C_0, C_1)) = (m, r_0, r_1)$.
- Given a decommitment $\pi = (m, r_0, r_1)$, $\textsc{Ver-Open}((N, U, \ell), m, \pi, (C_0, C_1))$ checks $C_0 = Ur_0^2$ and $C_1 = C_0^m r_1^{2^\ell}$.

The correctness of this commitment scheme follows directly from the above definitions. Simulatability follows if we simply let the simulator set up U as $U = y^{2^\ell}$. The details of the simulator are in the full version of the paper.

We have only to show that this scheme is binding. Suppose there exists a hard commitment (C_0, C_1) which can be opened as (m, r_0, r_1), and (m', r_0', r_1'), where $m = b_1 \ldots b_\ell$, and $m' = b_1' \ldots b_\ell'$. Both openings can be successfully verified, thus we have $C_0 = Ur_0^2 = Ur_0'^2$, and $C_1 = C_0^m r_1^{2^\ell} = C_0^{m'} r_1'^{2^\ell}$. Given that $m \neq m'$, this means that $r_1 \neq r_1'$. Let $f_0(x) = x^2$, $f_1(x) = C_0 x^2$. Note that finding a claw (i.e. x_0, x_1 such that $f_0(x_0) = f_1(x_1)$) would give a square root of U: $(U = (x_0 x_1^{-1} r_0)^2)$. This would then allow us to factor N. Thus, this is a claw-free pair. Note also that $C_0^m r_1^{2^\ell} = f_{b_\ell}(f_{b_{\ell-1}}(\ldots(f_{b_1}(r_1))))$. Since there are two verifiable openings, this must be equal to $C_0^{m'} r_1'^{2^\ell} = f_{b_\ell'}(f_{b_{\ell-1}'}(\ldots(f_{b_1'}(r_1'))))$. Let i be the smallest index such that $f_{b_i}(f_{b_{i-1}}(\ldots(f_{b_1}(r_1)))) = f_{b_i'}(f_{b_{i-1}'}(\ldots(f_{b_1'}(r_1'))))$. Such an i must clearly exist, and as long as $r_1' \neq r_1$ we also have $b_i \neq b_i'$. Thus we have found a claw between f_0 and f_1 which will allow us to factor N.

A similar proof shows that we cannot tease-open to one value and hard open to another.

Note that a similar scheme using an arbitrary RSA modulus N can be created using a modified version of the commitment described in [Fis01].

3 Constructing Zero-Knowledge Sets

3.1 Definition of Zero-Knowledge Sets

Let us start with an informal definition. A zero-knowledge set scheme (more generally, an elementary database scheme) [MRK03] consists of three algorithms, ZKS-SETUP, P (the Prover) and V (the Verifier) such that three properties hold: (1) completeness: for any database D, for any x such that $D(x) = v$ (where v can be a value if $x \in D$ or $\perp$ if $x \notin D$) an honest Prover who correctly commits to D can always convince the verifier that $D(x) = v$; (2) soundness: once a commitment to the database D has been formed (even by a malicious Prover), no P' can, for the same x, convince the Verifier that $D(x) = v_1$ and $D(x) = v_2$ for $v_1 \neq v_2$; (3) zero-knowledge: there exists a simulator S such that even for adversarially chosen D, no adversarial verifier can tell whether he is (a) talking to an honest prover P committed to D, or (b) talking to S who only has oracle access to the data set D.

The formalization of this definition, which is a slight revision of the original definition of [MRK03] (in particular, it extends the original definition to allow computational zero-knowledge), is given in the full version of the paper.

3.2 ZKS from Mercurial Commitments

In this section we show how, given a mercurial commitment scheme and a collision-resistant hash function, we can construct a zero-knowledge set. As already mentioned, this construction is essentially the same as the construction of [MRK03] with the mercurial commitments abstracted as a building block.

On the Role of Collision-Resistant Hashing. In order to construct ZKS from mercurial commitments, we need an additional property: that an ordered pair of commitments produced by HARD-COMM$(PK, \cdot, \cdot)$ and/or SOFT-COMM$(PK, \cdot)$ is in the domain D_{PK} of the commitment scheme (this property is needed because we will build trees of commitments, with the parent committing to its two children). This can be accomplished for any mercurial commitment scheme with sufficiently large D_{PK} by combining it with a collision-resistant hash function H that maps pairs of commitments into D_{PK}. Then, to commit to a pair of commitments, one would simply commit to its hash value instead. This approach was already used by [MRK03] with the DL-based mercurial commitment scheme implicitly constructed there.

The key for the hash function can be included as part of the commitment scheme's public key. The security of the resulting construction is easy to prove (we will not do so here for lack of space and because the arguments are standard). From now on, in describing the ZKS construction from mercurial commitments, we will assume that domain of a mercurial commitment scheme includes pairs of commitments.

We note that it is *necessary* to assume collision-resistant hash functions (CRHFs) because they are implied by ZKS: to build CRHF from ZKS, simply run the prover algorithm on the database D to produce a commitment C,

fixing the prover-randomness to an all-0 string. We can view an arbitrary-length string $b_1 b_2 \ldots b_\ell$ as an elementary database D where $D(i) = b_i$ for $1 \leq i \leq \ell$, and $i > \ell$ is not in the database. It is easy to see that the resulting algorithm is collision-resistant: an adversary who could produce two strings (databases) that hash to the same commitment C would contradicts the soundness property of ZKS.

Building ZKS. Below, we construct ZKS for a database D with keys of length[5] l given

- a mercurial commitment scheme (SETUP, HARD-COMM, SOFT-COMM, TEASE, OPEN, VER-TEASE, VER-OPEN) whose domain includes the values v contained in the database, as well as pairs of commitments (produced by HARD-COMM and/or SOFT-COMM);
- a pseudorandom function family $\{F_s\}_{s \in S}$ that maps binary string of length up to l to binary strings of length needed for random inputs r to HARD-COMM and SOFT-COMM.[6]

Our construction will be in the shared random string model if the mercurial commitment scheme (and the collision-resistant hash function, if separate from the mercurial commitment scheme) both require no more than a shared random string for their parameters. Otherwise, it will be in the trusted parameters model.

Intuition Informally, to generate a commitment *com* to the database D, the prover views each key x as an integer numbering a leaf of a height-l binary tree, and places a commitment to the information $v = D(x)$ into leaf number x. Each internal node of the tree is generated to contain the commitment to the contents of its two children. The result is thus a Merkle tree, except that each internal node is a commitment to, rather than a hash of, its two children. The value *com* is then the value contained in the root of the tree.

To respond to a query about x, the prover simply decommits the corresponding leaf and provides the authenticating path (together with all the decommitments) to the root.

The only problem with the just-described approach is that it requires exponential time (in l) to compute the tree, because the tree has 2^l leaves. This is where mercurial commitments help. Our exponential-size Merkle-like tree will have large subtrees where every leaf is a commitment to $\perp$, because the corresponding keys are not in the database. Instead of actually computing such a subtree ahead of time, the prover simply forms the root of this subtree as a soft-commitment, and does not do anything for other nodes of the subtree. Thus,

[5] As suggested in [MRK03], we can apply collision-resistant hashing to the keys if they are longer, although we will give up perfect completeness if two keys x_1 and x_2 such $D(x_1) \neq D(x_2)$ hash to the same result.

[6] The pseudorandom function family is needed only to save prover storage and make the prover stateless; if the prover is willing to store the amount of randomness proportional to $l(|D| + q)$, where q is the number of queries, then it is not necessary.

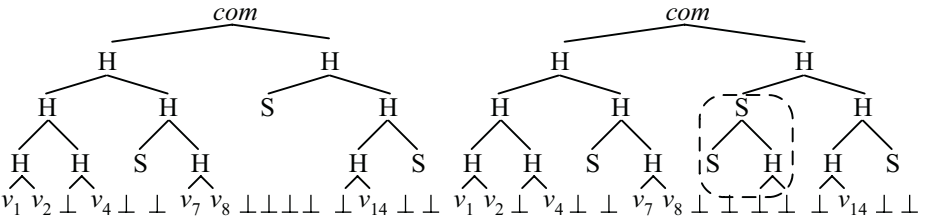

Fig. 1. A commitment tree before and after a query for key 11, whose value is not the database. The parts built in response to the query are shown in the second tree. Hard commitments are denoted by 'H' and soft commitments by 'S'. Each leaf contains a commitment to the value shown rather than the value itself

the resulting Merkle tree gets "pruned" to size at most $2l|D|$, because the only nodes in the tree are ancestors of leaves in D and their siblings. (This is because if a node is neither an ancestor of a leaf in D nor a sibling of such an ancestor, then both its and its sibling's subtrees are empty.)

Answering queries about $x \in D$ is still done the same way. In response to queries about $x \notin D$ the prover generates the portion of the subtree that is missing (from x to the root of the empty subtree). The value at the root of the empty subtree is then teased (soft-decommitted) to its two (newly generated) children, and the entire authenticating path from x to *com* is provided using teasers, rather than hard decommitments. This is illustrated in Figure 1.

To save the prover from the expense of having to remember all the choices it made when generating the tree (both initially and in response to $x \notin D$ queries), we generate all random strings used in the commitments pseudorandomly rather than randomly.

Soundness follows from the fact that soft decommitments always have the same semantics, namely that $x \notin D$, and that soft decommitments cannot, by the binding property, disagree with hard decommitments. Zero-knowledgeness follows from the simulatability of commitments and from the fact that decommitments are consistent: a given node will never be (hard- or soft-) decommitted in two different ways. Note that zero-knowledge will be perfect, statistical, or computational, depending on the simulatability of mercurial commitments (however, for perfect and statistical zero-knowledge, the prover must use truly random, rather than pseudorandom, strings; hence, it must be stateful in order to remember the random strings it used when responding to queries.)

We formalize the above description in the full version of the paper.

Improving the Efficiency of DL-Based Construction. While in general any collision-resistant hash function can be used with our DL-based mercurial commitments, Pedersen's hash function [Ped92] is suggested by [MRK03] because it is also based on the discrete logarithm assumption and its parameters can be selected from a shared random string. Given a group G of prime order q and two generators g and h, Pedersen's hash function $H_{G,g,h}$ hashes two integers

$0 \leq a, b < q$ into a single element $h \in G$ via $h = g^a h^b$. It is easy to see that this hash function is collision-resistant if discrete logarithms in G are hard.

It may seem that Pedersen's hash function is well suited for use with DL-based mercurial commitments over the same group G. This, however, is not true, because the range of hash function is G, while the domain of the commitments is $\{0, 1, \ldots, q-1\}$. In particular, if G is the subgroup of size q of $\mathbb{Z}_p^*$ for $q|(p-1)$, one would need to choose two separate sets of parameters: (q_1, p_1) for the hash function and (q_2, p_2) for the commitment scheme, with $q_2 \geq p_1$. This seems to be necessary for the construction of [MRK03] to work (although it is not explicitly stated there).

In addition, to hash two commitments (which consist of two elements of $\mathbb{Z}_{p_2}^*$ each), multiple iterations of the hash function are needed, because a single iteration can handle only a pair of elements of $\mathbb{Z}_{q_1}$.

Here, we point out two minor modifications the Pedersen's hash function that eliminate the need for a second set of parameters and minimize the number of iterations necessary to hash two commitments. Both modifications rely on folklore results.

First, we will modify the hash function to take four inputs instead of two by using four generators (all in the shared random string), g_1, g_2, g_3, g_4, and outputting, on input (a, b, c, d), the value $g_1^a g_2^b g_3^c g_4^d$. The proof of collision-resistance of this function is a simple exercise and is omitted here.

Our second modification relies (in addition to the DL assumption) on the assumption that Sophie Germain primes are sufficiently dense (recall that a q is a Sophie Germain prime if $p = 2q + 1$ is prime). We let q be a Sophie Germain prime. Then the subgroup G of order q of $\mathbb{Z}_p^*$ is QR_p. Consider the following efficient bijection ϕ' between QR_p and $\{1, 2, \ldots, q\}$: if $x \leq q$, $\phi'(x) = x$; else $\phi'(x) = p - x$ (this is a bijection because exactly one of $(x, -x)$ is in QR_p, because $p \equiv 3 \pmod 4$ since q is odd). Now let $\phi(x) = \phi'(x) - 1$ to make the range of the bijection $\{0, 1, \ldots, q-1\}$.

The bijection ϕ allows us to view $G = QR_p$ and $\{0, 1, \ldots, q-1\}$ as essentially the same.[7] Thus, we will simply modify Pedersen's hash function to output $\phi(h)$ instead of h, and to take inputs in QR_p instead of $\{0, 1, \ldots, q-1\}$ by applying ϕ^{-1} to them first.

The resulting ZKS scheme takes seven values and seven exponentiations per level of the hash tree to verify (four for the hash function and three for the mercurial commitment). Note that two of the generators can be shared between the hash function and the mercurial commitment scheme. Because verifications require only products of fixed-base exponentiations with four bases (except in the case of tease verification, when a single exponentiation with a random base is required), much precomputation can be done to speed up verification (see, e.g., [LL94], which can be extended to multiple bases).

[7] We remark that, obviously, a bijection between G and $\{0, 1, \ldots, q-1\}$ always exists because $|G| = q$; the reason for using Sophie Germain primes is that we do not know how construct a simple efficient bijection otherwise.

Acknowledgments

The authors wish to thank Rosario Gennaro and Adi Shamir for helpful technical discussions, and Monica Perez for giving our new commitments their name "mercurial."

The authors are supported in part by the following: Melissa Chase by NSF Grant CNS-0347661 and NSF Graduate Research Fellowship; Alexander Healy by NSF Grant CCR-0205423 and a Sandia Fellowship; Anna Lysyanskaya by NSF Grant CNS-0374661; Tal Malkin by NSF Grant CCF-0347839; and Leonid Reyzin by NSF Grant CCR-0311485.

References

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2), 1988.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Guiseppe Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6), 1991.

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: non-interactive zero-knowledge based on any trapdoor permutation. *J. Cryptology*, 9(3), 1996.

[Fis01] Marc Fischlin. *Trapdoor Commitment Schemes and Their Applications*. PhD thesis, University of Frankfurt am Main, December 2001.

[FLS99] U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Computing*, 29(1), 1999.

[GM05] Rosario Gennaro and Silvio Micali. Independent zero-knowledge sets. Unpublished manuscript, 2005.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2), 1988.

[GO92] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In *CRYPTO '92*.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Computing*, 28(4), 1999.

[LL94] Chae Hoon Lim and Pil Joong Lee. More flexible exponentiation with precomputation. In *Advances in Cryptology—CRYPTO '94*.

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *Advances in Cryptology — CRYPTO 2002*.

[MRK03] Silvio Micali, Michael Rabin, and Joe Kilian. Zero-knowledge sets. In *Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

[MRV99] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *Proc. 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1999.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):51–158, 1991.

[ORS04] Rafi Ostrovsky, Charles Rackoff, and Adam Smith. Efficient consistency proof on a committed database. In *Proc. of Automata, Languages and Programming: 31st International Colloquium, ICALP 2004*.

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91*.

Hierarchical Identity Based Encryption with Constant Size Ciphertext*

Dan Boneh[1,**], Xavier Boyen[2], and Eu-Jin Goh[1,**]

[1] Computer Science Department, Stanford University
{dabo, eujin}@cs.stanford.edu
[2] Voltage Inc., Palo Alto
xb@boyen.org

Abstract. We present a Hierarchical Identity Based Encryption (HIBE) system where the ciphertext consists of just three group elements and decryption requires only two bilinear map computations, *regardless of the hierarchy depth*. Encryption is as efficient as in other HIBE systems. We prove that the scheme is selective-ID secure in the standard model and fully secure in the random oracle model. Our system has a number of applications: it gives very efficient forward secure public key and identity based cryptosystems (with short ciphertexts), it converts the NNL broadcast encryption system into an efficient public key broadcast system, and it provides an efficient mechanism for encrypting to the future. The system also supports limited delegation where users can be given restricted private keys that only allow delegation to bounded depth. The HIBE system can be modified to support sublinear size private keys at the cost of some ciphertext expansion.

1 Introduction

An Identity Based Encryption (IBE) system [24, 5] is a public key system where the public key can be an arbitrary string such as an email address. A central authority uses a master key to issue private keys to identities that request them. Hierarchical IBE (HIBE) [17, 14] is a generalization of IBE that mirrors an organizational hierarchy. An identity at level k of the hierarchy tree can issue private keys to its descendant identities, but cannot decrypt messages intended for other identities (details are given in Section 2.1). The first construction for HIBE is due to Gentry and Silverberg [14] where security is based on the Bilinear Diffie-Hellman (BDH) assumption in the random oracle model. A subsequent construction due to Boneh and Boyen [1] gives an efficient (selective-ID secure) HIBE based on BDH without random oracles. In both constructions, the length of ciphertexts and private keys, as well as the time needed for decryption and encryption, grows linearly in the depth ℓ of the hierarchy.

* Extended Abstract. Full version available on the Cryptology ePrint Archives [3].
** Supported by NSF.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 440–456, 2005.

There are currently two principal applications for HIBE. The first, due to Canetti, Halevi, and Katz [9], is forward secure encryption. Forward secure encryption enables users to periodically update their private keys so that a message encrypted at period n cannot be read using a private key from period $n' > n$. To provide for $T = 2^t$ time periods, the CHK construction uses a HIBE of depth t where identities are *binary* vectors of length at most t. At time n, the encryptor encrypts using the identity corresponding to the n-th node of this depth t binary tree. Consequently, using previous HIBE systems [14, 1], ciphertexts in this forward secure construction are of size $O(t)$; private keys are of size $O(t^2)$ but can be reduced to size $O(t)$ by using updateable public storage. The second application for HIBE, due to Dodis and Fazio [11], is using HIBE to convert the NNL broadcast encryption system [22] into a *public-key* broadcast system. Unfortunately, the resulting public-key broadcast system is no better than simpler constructions because ciphertext length in previous HIBE constructions is linear in the depth of the hierarchy.

Our Contribution. We present a HIBE system where the ciphertext size as well as the decryption cost are *independent* of the hierarchy depth ℓ. Ciphertexts in our HIBE system are always just three group elements and decryption requires only two bilinear map computations. Private keys in our basic system contain ℓ group elements as in previous HIBE constructions.

Our system gives a forward secure encryption system with short ciphertexts consisting of only three group elements, for any number $T = 2^t$ of time periods. With our basic HIBE system, the private key size in this forward secure encryption system is $O(t^2)$. In Section 4 we describe a hybrid system that borrows some features from the Boneh-Boyen HIBE [1] and results in a forward secure encryption scheme where private key size is reduced to $O(t^{3/2})$ and ciphertext size is $O(\sqrt{t})$. By using updateable public storage as in CHK [9], private key size in these systems can be further reduced to size $O(t)$ and $O(\sqrt{t})$ respectively. In addition, instantiating the Dodis-Fazio [11] system with our HIBE system results in a *public-key* broadcast system that is as efficient as the NNL subset difference method.

It is worth noting that private keys in our system *shrink* as the identity depth increases; this shrinkage is the opposite behavior from previous HIBE systems where private keys become larger as we descend deeper down the hierarchy tree. This behavior leads to "limited delegation" where an identity at depth k can be given a restricted private key that only lets it issue private keys to descendants of limited depth (as opposed to any descendant).

Security of our system is based on a natural assumption that is closely related to the Diffie-Hellman Inversion assumption [1, 19]. We describe the assumption in Section 2.3. In the full paper [3], we prove a lower bound on the computational complexity of the problem in the generic group model and also discuss its relation to existing assumptions in bilinear groups. We present the system in Section 3 and prove its security in the selective identity model without using random oracles. We then observe that a selective-ID secure HIBE results in a fully secure HIBE in the random oracle model. In Sections 4 and 5 we discuss

several extensions and applications of the system. For example, in addition to the applications already mentioned, we show how private keys can be further compressed to sublinear size and also describe an efficient mechanism for encrypting to the future.

2 Preliminaries

We briefly review the definition of HIBE and bilinear groups, and introduce the Bilinear Diffie-Hellman Exponent assumption in such groups.

2.1 Fully Secure HIBE Systems

Like an Identity Based Encryption (IBE) system, a Hierarchical Identity Based Encryption (HIBE) system consists of four algorithms [17, 14, 1]: `Setup`, `KeyGen`, `Encrypt`, `Decrypt`. In HIBE, however, identities are vectors; a vector of dimension k represents an identity at depth k. The `Setup` algorithm generates system parameters, denoted by *params*, and a master key *master-key*. We refer to the *master-key* as the private key at depth 0 and note that an IBE system is a HIBE where all identities are at depth 1. Algorithm `KeyGen` takes as input an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ at depth k and the private key $d_{\mathsf{ID}|k-1}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1})$ at depth $k-1$, and then outputs the private key d_{ID} for identity ID. The encryption algorithm encrypts messages for an identity using *params* and the decryption algorithm decrypts ciphertexts using the private key.

Chosen ciphertext security for HIBE systems is defined under a chosen identity attack where the adversary is allowed to adaptively chose the public key on which it will be challenged. More precisely, HIBE security (IND-ID-CCA) is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Setup: The challenger $\mathcal{C}$ runs the `Setup` algorithm and gives $\mathcal{A}$ the resulting system parameters *params*, keeping the *master-key* to itself.

Phase 1: $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_m$ where query q_i is one of the following:

- Private key query $\langle \mathsf{ID}_i \rangle$. $\mathcal{C}$ responds by running algorithm `KeyGen` to generate the private key d_i corresponding to the public key $\langle \mathsf{ID}_i \rangle$ and sends d_i to $\mathcal{A}$.
- Decryption query $\langle \mathsf{ID}_i, C_i \rangle$. $\mathcal{C}$ responds by running algorithm `KeyGen` to generate the private key d corresponding to ID_i. It then runs algorithm `Decrypt` to decrypt the ciphertext C_i using the private key d and sends the resulting plaintext to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs an identity ID^* and two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ on which it wishes to be challenged. The only restriction is that $\mathcal{A}$ did not previously issue a private key query for ID^* or a prefix of ID^*. $\mathcal{C}$ picks a random bit $b \in \{0, 1\}$ and sets the challenge ciphertext to $\mathsf{CT} = \mathtt{Encrypt}(\mathit{params}, \mathsf{ID}^*, M_b)$, which is sent to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues additional queries $q_{m+1}, \ldots, q_n$ where q_i is one of:

- Private key query $\langle \mathsf{ID}_i \rangle$ where $\mathsf{ID}_i \neq \mathsf{ID}^*$ and ID_i is not a prefix of ID^*.
- Decryption query $\langle C_i \rangle \neq \langle C \rangle$ for ID^* or any prefix of ID^*.

In both cases, $\mathcal{C}$ responds as in Phase 1. These queries may be adaptive.

Guess: Finally, the adversary outputs a guess $b' \in \{0, 1\}$ and wins if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}$ as

$$\mathrm{Adv}_{\mathcal{E},\mathcal{A}} = |\Pr[b = b'] - 1/2| \,.$$

The probability is over the random bits used by the challenger and the adversary.

Canetti, Halevi, and Katz [9, 10] define a weaker notion of security in which the adversary commits ahead of time to the public key it will attack. We refer to this notion as selective identity, chosen ciphertext secure HIBE (IND-sID-CCA). The game is exactly the same as IND-ID-CCA except that the adversary $\mathcal{A}$ discloses to the challenger the target identity ID^* before the **Setup** phase. The restrictions on private key queries from phase 2 also hold in phase 1.

Definition 1. *We say that a HIBE system $\mathcal{E}$ is $(t, q_{ID}, q_C, \epsilon)$-secure if for any t-time* IND-ID-CCA *(respectively* IND-sID-CCA*) adversary $\mathcal{A}$ that makes at most q_{ID} chosen private key queries and at most q_C chosen decryption queries, we have that $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is $(t, q_{ID}, q_C, \epsilon)$-*IND-ID-CCA *(resp.* IND-sID-CCA*) secure.*

Semantic Security. As usual, we define chosen plaintext security for a HIBE system as in the preceding game, except that the adversary is not allowed to issue any decryption queries. The adversary may still issue adaptive private key queries. This security notion is termed as IND-ID-CPA (or IND-sID-CPA in the case of a selective identity adversary).

Definition 2. *We say that a HIBE system $\mathcal{E}$ is (t, q_{ID}, ϵ)-*IND-ID-CPA *secure (resp.* IND-sID-CPA*) if $\mathcal{E}$ is $(t, q_{ID}, 0, \epsilon)$-*IND-ID-CCA *secure (resp.* IND-sID-CCA*).*

2.2 Bilinear Groups

We briefly review bilinear maps and bilinear map groups. We use the following notation [18, 8]:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of prime order p;
2. g is a generator of $\mathbb{G}$.
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$.

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two groups as above. A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ with the properties:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists both a group $\mathbb{G}_1$ and an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ as above.

2.3 Bilinear Diffie-Hellman Exponent (BDHE) Assumption

The ℓ-BDHE problem in $\mathbb{G}$ is as follows: given g, h, and $g^{(\alpha^i)}$ in $\mathbb{G}$ for $i = 1, 2, \ldots, \ell-1, \ell+1, \ldots, 2\ell$ as input, output $e(g,h)^{(\alpha^\ell)} \in \mathbb{G}_1$. Since $g^{(\alpha^\ell)}$ is missing from the list of powers, the bilinear map seems to be of no help in computing $e(g,h)^{(\alpha^\ell)}$. As a shorthand, let $y_i = g^{(\alpha^i)} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving ℓ-BDHE in $\mathbb{G}$ if

$$\Pr\left[\mathcal{A}\big(g, h, y_1, \ldots, y_{\ell-1}, y_{\ell+1}, \ldots, y_{2\ell}\big) = e(g,h)^{(\alpha^\ell)}\right] \geq \epsilon,$$

where the probability is over the random choice of generators g, h in $\mathbb{G}$, the random choice of α in $\mathbb{Z}_p$, and the random bits used by $\mathcal{A}$. The decisional version of the ℓ-BDHE problem in $\mathbb{G}$ is defined in the usual manner. Let $\overrightarrow{y}_{g,\alpha,\ell} = (y_1, \ldots, y_{\ell-1}, y_{\ell+1}, \ldots, y_{2\ell})$. An algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ϵ in solving decision ℓ-BDHE in $\mathbb{G}$ if

$$\left|\Pr\left[\mathcal{B}\big(g, h, \overrightarrow{y}_{g,\alpha,\ell}, e(g,h)^{(\alpha^\ell)}\big) = 0\right] - \Pr\left[\mathcal{B}\big(g, h, \overrightarrow{y}_{g,\alpha,\ell}, T\big) = 0\right]\right| \geq \epsilon,$$

where the probability is over the random choice of generators g, h in $\mathbb{G}$, the random choice of α in $\mathbb{Z}_p$, the random choice of $T \in \mathbb{G}_1$, and the random bits consumed by $\mathcal{B}$. We refer to the distribution on the left as $\mathcal{P}_{BDHE}$ and the distribution on the right as $\mathcal{R}_{BDHE}$.

Definition 3. *We say that the (decision) (t, ϵ, ℓ)-BDHE assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the (decision) ℓ-BDHE problem in $\mathbb{G}$.*

For conciseness we occasionally drop the t and ϵ and simply refer to the (decision) ℓ-BDHE in $\mathbb{G}$. In the full version of this paper [3], we show that a broad class of assumptions, including the ℓ-BDHE assumption, hold in generic bilinear groups [25]; we also discuss the relation between these assumptions. We show that the ℓ-BDHE is a natural extension of the Bilinear Diffie-Hellman Inversion problem, which was previously used in various constructions [1, 12, 19].

3 A HIBE System with Constant Size Ciphertext

Let $\mathbb{G}$ be a bilinear group of prime order p and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a bilinear map. For now, we assume that public keys (that is, identities ID) at depth k are vectors of elements in $(\mathbb{Z}_p^*)^k$. We write $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$. The j-th component corresponds to the identity at level j. We later extend the construction to public keys over $\{0,1\}^*$ by first hashing each component I_j using

a collision resistant hash $H : \{0,1\}^* \to \mathbb{Z}_p^*$. We also assume that the messages to be encrypted are elements in $\mathbb{G}_1$. The HIBE system works as follows:

Setup(ℓ): To generate system parameters for an HIBE of maximum depth ℓ, select a random generator $g \in \mathbb{G}$, a random $\alpha \in \mathbb{Z}_p$, and set $g_1 = g^\alpha$. Next, pick random elements $g_2, g_3, h_1, \ldots, h_\ell \in \mathbb{G}$. The public parameters and the master key are

$$params = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell) , \qquad master\text{-}key = g_2^\alpha.$$

KeyGen($d_{\mathsf{ID}|k-1}, \mathsf{ID}$): To generate a private key d_{ID} for identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ of depth $k \leq \ell$, pick a random $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{ID}} = \left(g_2^\alpha \cdot \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^r, \quad g^r, \quad h_{k+1}^r, \quad \ldots, \quad h_\ell^r \right) \in \mathbb{G}^{2+\ell-k}.$$

Note that d_{ID} becomes shorter as the depth of ID increases. The private key for ID can be generated just given a private key for $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1}) \in (\mathbb{Z}_p^*)^{k-1}$ as required. Indeed, let

$$d_{\mathsf{ID}|k-1} = \left(g_2^\alpha \cdot \left(h_1^{\mathrm{I}_1} \cdots h_{k-1}^{\mathrm{I}_{k-1}} \cdot g_3\right)^{r'}, \quad g^{r'}, \quad h_k^{r'}, \ldots, h_\ell^{r'} \right) = (a_0, a_1, b_k, \ldots, b_\ell)$$

be the private key for $\mathsf{ID}_{|k-1}$. To generate d_{ID}, pick a random $t \in \mathbb{Z}_p$ and output

$$d_{\mathsf{ID}} = \left(a_0 \cdot b_k^{\mathrm{I}_k} \cdot \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^t, \quad a_1 \cdot g^t, \quad b_{k+1} \cdot h_{k+1}^t, \quad \ldots, \quad b_\ell \cdot h_\ell^t \right).$$

This private key is a properly distributed private key for $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ for $r = r' + t \in \mathbb{Z}_p$.

Encrypt(*params*, ID, M): To encrypt a message $M \in \mathbb{G}_1$ under the public key $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$, pick a random $s \in \mathbb{Z}_p$ and output

$$\mathsf{CT} = \left(e(g_1, g_2)^s \cdot M, \quad g^s, \quad \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^s \right) \in \mathbb{G}_1 \times \mathbb{G}^2.$$

Decrypt($d_{\mathsf{ID}}, \mathsf{CT}$): Consider an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$. To decrypt a given ciphertext $\mathsf{CT} = (A, B, C)$ using the private key $d_{\mathsf{ID}} = (a_0, a_1, b_{k+1} \ldots, b_\ell)$, output

$$A \cdot e(a_1, C) \Big/ e(B, a_0) = M.$$

Indeed, for a valid ciphertext, we have

$$\frac{e(a_1, C)}{e(B, a_0)} = \frac{e\left(g^r,\ (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3)^s\right)}{e\left(g^s,\ g_2^\alpha (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3)^r\right)} = \frac{1}{e(g, g_2)^{s\alpha}} = \frac{1}{e(g_1, g_2)^s}.$$

Observe that for identities at any depth, the ciphertext contains only 3 elements and decryption takes only 2 pairings. In previous HIBE systems, ciphertext size and decryption time grow linearly in the identity depth. Also, note that $e(g_1, g_2)$ used for encryption can be precomputed (or substituted for g_2 in the system parameters) so that encryption does not require any pairings.

3.1 Security

We first show that our HIBE scheme is selective identity secure (IND-sID-CPA) under the decisional Bilinear Diffie-Hellman Exponent assumption. We later describe how to provide both chosen ciphertext security (IND-sID-CCA) and full HIBE security (IND-ID-CCA).

Theorem 1. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the decision $(t, \epsilon, \ell+1)$-BDHE assumption holds in $\mathbb{G}$. Then the previously defined ℓ-HIBE system is (t', q_S, ϵ)-selective identity, chosen plaintext* (IND-sID-CPA) *secure for arbitrary ℓ, q_S, and $t' < t - \Theta(\tau\, \ell\, q_S)$, where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the ℓ-HIBE system. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the decision $(\ell+1)$-BDHE problem in $\mathbb{G}$.

For a generator $g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$ let $y_i = g^{(\alpha^i)} \in \mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random tuple $(g, h, y_1, \ldots, y_\ell, y_{\ell+2}, \ldots, y_{2\ell+2}, T)$ that is either sampled from $\mathcal{P}_{BDHE}$ (where $T = e(g,h)^{(\alpha^{\ell+1})}$) or from $\mathcal{R}_{BDHE}$ (where T is uniform and independent in $\mathbb{G}_1$). Algorithm $\mathcal{B}$'s goal is to output 1 when the input tuple is sampled from $\mathcal{P}_{BDHE}$ and 0 otherwise. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows:

Initialization. The selective identity game begins with $\mathcal{A}$ first outputting an identity $\mathsf{ID}^* = (\mathrm{I}_1^*, \ldots, \mathrm{I}_m^*) \in (\mathbb{Z}_p^*)^m$ of depth $m \le \ell$ that it intends to attack. If $m < \ell$ then $\mathcal{B}$ pads ID^* with $\ell - m$ zeroes on the right to make ID^* a vector of length ℓ. Hence, from here we assume that ID^* is a vector of length ℓ.

Setup. To generate the system parameters, algorithm $\mathcal{B}$ picks a random γ in $\mathbb{Z}_p$ and sets $g_1 = y_1 = g^\alpha$ and $g_2 = y_\ell \cdot g^\gamma = g^{\gamma+(\alpha^\ell)}$. Next, $\mathcal{B}$ picks random $\gamma_1, \ldots, \gamma_\ell$ in $\mathbb{Z}_p$ and sets $h_i = g^{\gamma_i}/y_{\ell-i+1}$ for $i = 1, \ldots, \ell$. Algorithm $\mathcal{B}$ also picks a random δ in $\mathbb{Z}_p$ and sets $g_3 = g^\delta \cdot \prod_{i=1}^{\ell} y_{\ell-i+1}^{I_i^*}$.

Finally, $\mathcal{B}$ gives $\mathcal{A}$ the system parameters $params = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell)$. Observe that all these values are distributed uniformly and independently in $\mathbb{G}$ as required. The master key corresponding to these system parameters is $g_2^\alpha = g^{\alpha(\alpha^\ell+\gamma)} = y_{\ell+1} y_1^\gamma$, which is unknown to $\mathcal{B}$ since $\mathcal{B}$ does not have $y_{\ell+1}$.

Phase 1. $\mathcal{A}$ issues up to q_S private key queries. Consider a query for the private key corresponding to $\mathsf{ID} = (\mathrm{I}_1, \ldots, I_u) \in (\mathbb{Z}_p^*)^u$ where $u \le \ell$. The only restriction is that ID is not ID^* or a prefix of ID^*. This restriction ensures that there exists a $k \in \{1, \ldots, u\}$ such that $\mathrm{I}_k \neq \mathrm{I}_k^*$ (otherwise, ID would be a prefix of ID^*). To respond to the query, algorithm $\mathcal{B}$ first derives a private key for the identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$ from which it then constructs a private key for the requested identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k, \ldots, \mathrm{I}_u)$.

To generate the private key for identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$, $\mathcal{B}$ first picks a random $\tilde{r}$ in $\mathbb{Z}_p$. We pose $r = \frac{\alpha^k}{(\mathrm{I}_k - \mathrm{I}_k^*)} + \tilde{r} \in \mathbb{Z}_p$. Next, $\mathcal{B}$ generates the private key

$$\left(g_2^\alpha \cdot (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} g_3)^r,\ \ g^r,\ \ h_{k+1}^r,\ \ \ldots,\ \ h_\ell^r \right), \tag{1}$$

which is a properly distributed private key for the identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$. We show that $\mathcal{B}$ can compute all elements of this private key given the values at its disposal. We use the fact that $y_i^{(\alpha^j)} = y_{i+j}$ for any i, j.

To generate the first component of the private key, first observe that

$$(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} g_3)^r = \left(g^{\delta + \sum_{i=1}^{k} \mathrm{I}_i \gamma_i} \cdot \prod_{i=1}^{k-1} y_{\ell-i+1}^{(\mathrm{I}_i^* - \mathrm{I}_i)} \cdot y_{\ell-k+1}^{(\mathrm{I}_k^* - \mathrm{I}_k)} \cdot \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{\mathrm{I}_i^*} \right)^r . \quad (2)$$

Let Z denote the product of the first, second, and fourth terms. That is,

$$Z = \left(g^{\delta + \sum_{i=1}^{k} \mathrm{I}_i \gamma_i} \cdot \prod_{i=1}^{k-1} y_{\ell-i+1}^{(\mathrm{I}_i^* - \mathrm{I}_i)} \cdot \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{\mathrm{I}_i^*} \right)^r .$$

One can verify that $\mathcal{B}$ can compute all the terms in Z given the values at its disposal. Next, observe that the third term in Eq (2), namely $y_{\ell-k+1}^{r(\mathrm{I}_k^* - \mathrm{I}_k)}$, is:

$$y_{\ell-k+1}^{r(\mathrm{I}_k^* - \mathrm{I}_k)} = y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)} \cdot y_{\ell-k+1}^{(\mathrm{I}_k^* - \mathrm{I}_k) \frac{\alpha^k}{(\mathrm{I}_k - \mathrm{I}_k^*)}} = y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)} / y_{\ell+1}.$$

Hence, the first component in the private key (1) is equal to:

$$g_2^{\alpha} (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} g_3)^r = (y_{\ell+1} y_1^{\gamma}) \cdot Z \cdot (y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)} / y_{\ell+1}) = y_1^{\gamma} \cdot Z \cdot y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)}.$$

Since $y_{\ell+1}$ cancels out, all the terms in this expression are known to $\mathcal{B}$. Thus, $\mathcal{B}$ can compute the first private key component.

The second component, g^r, is $y_k^{1/(\mathrm{I}_k - \mathrm{I}_k^*)} g^{\tilde{r}}$ which $\mathcal{B}$ can compute. Similarly, the remaining elements $h_{k+1}^r, \ldots, h_\ell^r$ can be computed by $\mathcal{B}$ since they do not involve a $y_{\ell+1}$ term. Thus, $\mathcal{B}$ can derive a valid private key for $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$. Algorithm $\mathcal{B}$ uses this private key to derive a private key for the descendant identity ID and gives $\mathcal{A}$ the result.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathbb{G}_1$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and responds with the challenge ciphertext

$$\mathsf{CT} = (M_b \cdot T \cdot e(y_1, h^{\gamma}),\ \ h,\ \ h^{\delta + \sum_{i=1}^{\ell} \mathrm{I}_i^* \gamma_i})$$

where h and T are from the input tuple given to $\mathcal{B}$. First note that if $h = g^c$ (for some unknown c in $\mathbb{Z}_p$) then

$$h^{\delta + \sum_{i=1}^{\ell} \mathrm{I}_i^* \gamma_i} = \left(\prod_{i=1}^{\ell} (g^{\gamma_i} / y_{\ell-i+1})^{\mathrm{I}_i^*} \cdot (g^{\delta} \prod_{i=1}^{\ell} y_{\ell-i+1}^{\mathrm{I}_i^*}) \right)^c = (h_1^{\mathrm{I}_1^*} \cdots h_\ell^{\mathrm{I}_\ell^*} g_3)^c, \quad \text{and}$$

$$e(g, h)^{(\alpha^{\ell+1})} \cdot e(y_1, h^{\gamma}) = \big(e(y_1, y_\ell) \cdot e(y_1, g^{\gamma}) \big)^c = e(y_1, y_\ell g^{\gamma})^c = e(g_1, g_2)^c.$$

Therefore, if $T = e(g, h)^{(\alpha^{\ell+1})}$ (i.e., when the input tuple is sampled from $\mathcal{P}_{BDHE}$), then the challenge ciphertext is a valid encryption of M_b under the

original (unpadded) identity $\mathsf{ID}^* = (\mathrm{I}_1^*, \ldots, \mathrm{I}_m^*)$ chosen by the adversary, since

$$\begin{aligned}\mathsf{CT} &= \left(M_b \cdot e(g_1, g_2)^c, \quad g^c, \quad (h_1^{\mathrm{I}_1^*} \cdots h_m^{\mathrm{I}_m^*} \cdots h_\ell^{\mathrm{I}_\ell^*} g_3)^c \right) \\ &= \left(M_b \cdot e(g_1, g_2)^c, \quad g^c, \quad (h_1^{\mathrm{I}_1^*} \cdots h_m^{\mathrm{I}_m^*} g_3)^c \right).\end{aligned}$$

On the other hand, when T is uniform and independent in $\mathbb{G}_1$ (when the input tuple is sampled from $\mathcal{R}_{BDHE}$), CT is independent of b in the adversary's view.

Phase 2. $\mathcal{A}$ issues queries not issued in Phase 1. $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $T = e(g, h)^{(\alpha^{\ell+1})}$. Otherwise, it outputs 0 meaning T is random in $\mathbb{G}_1$.

When the input tuple is sampled from $\mathcal{P}_{BDHE}$ (where $T = e(g, h)^{(\alpha^{\ell+1})}$), then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ satisfies $|\Pr[b = b'] - 1/2| \geq \epsilon$. When the input tuple is sampled from $\mathcal{R}_{BDHE}$ (where T is uniform in $\mathbb{G}_1$) then $\Pr[b = b'] = 1/2$. Therefore, with g, h uniform in $\mathbb{G}$, α uniform in $\mathbb{Z}_p$, and T uniform in $\mathbb{G}_1$ we have that

$$\left| \Pr\left[\mathcal{B}\big(g, h, \overrightarrow{y}_{g,\alpha,\ell}, e(g, h)^{(\alpha^{\ell+1})}\big) = 0\right] \quad - \quad \Pr\left[\mathcal{B}\big(g, h, \overrightarrow{y}_{g,\alpha,\ell}, T\big) = 0\right] \right| \geq \quad |(1/2 \pm \epsilon) - 1/2| \;=\; \epsilon$$

as required. This completes the proof of the theorem.

Chosen Ciphertext Security. Canetti et al. [10] show a general method of building an IND-sID-CCA secure ℓ-HIBE from a IND-sID-CPA secure $\ell+1$-HIBE. A more efficient construction is given by Boneh and Katz [7]. Applying either method to our HIBE construction results in a IND-sID-CCA secure ℓ-HIBE for arbitrary ℓ where the ciphertext length is independent of the hierarchy height.

Arbitrary Identities. We can extend our HIBE to handle arbitrary identities $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_\ell)$ with $\mathrm{I}_i \in \{0, 1\}^*$ for $i = 1, \ldots, \ell$ by hashing each I_i with a collision resistant hash function $H : \{0, 1\}^* \to \mathbb{Z}_p^*$ during key generation and encryption. A standard argument shows that if the original HIBE scheme is IND-sID-CCA secure, then so is the HIBE scheme using H.

3.2 Full HIBE Security

Theorem 1 shows that our HIBE system is selective-ID secure without random oracles. Thus, the system is secure when the adversary commits ahead of time to the identity he intends to attack. Boneh and Boyen [1] observed that IBE systems that are selective-ID secure are also fully secure (i.e., secure against adversaries that adaptively select the identity to attack) as long as one hashes the identity prior to using it. The reduction, however, is not tight. Let $H : \{0, 1\}^* \to \{0, 1\}^d$ be a hash function (where, e.g., $d = 160$ bits). Assuming H is collision resistant, the reduction introduces a 2^d multiplicative security loss factor in the standard

model. When H is viewed as a random oracle, the reduction introduces a q_H multiplicative security loss factor where q_H is the number of the hash oracle queries.

A similar observation applies to HIBE systems. Let $\mathcal{E}$ be a selective-ID secure HIBE of depth ℓ. Let $\mathcal{E}_H$ be an HIBE system where an identity $\mathsf{ID} = (I_1, \ldots, I_k)$ is hashed to $\mathsf{ID}_H = (H(I_1), \ldots, H(I_k))$ before using it in `KeyGen` and `Encrypt`. Then, if H is collision resistant, it follows that $\mathcal{E}_H$ is a fully secure HIBE, but the reduction introduces a loss factor of $2^{\ell d}$. In the random oracle model, $\mathcal{E}_H$ is a fully secure HIBE and the reduction introduces a loss factor of q_H^{ℓ}.

We remark that in the random oracle model, the public parameters are of constant size and contain only the two group elements (g, g_1); the other parameters $(g_2, g_3, h_1, \ldots, h_\ell)$ need not be specified as they can be derived by applying the oracle on a predetermined input string.

We also note that the construction of Waters [26], for a fixed depth ℓ, applied to our HIBE could give a constant ciphertext HIBE with a polynomial time reduction to the underlying complexity assumption. The resulting private keys are much larger, namely of size $d\ell$, as opposed to ℓ in our system.

4 Extensions

We discuss a number of extensions to the HIBE system of the previous section.

4.1 Limited Delegation

Let $d_{\mathsf{ID}} = (a_0, a_1, b_k, \ldots, b_\ell)$ be the private key for the identity ID. Note that the `Decrypt` algorithm uses only the terms a_0 and a_1, and the `KeyGen` algorithm uses only the remaining terms $b_k, \ldots, b_\ell$.

By removing any number of $b_k, \ldots, b_\ell$, an identity ID at depth k can be given a restricted private key that only lets it issue private keys to descendants of bounded depth. For example, if the private key for ID only contains b_k, b_{k+1}, b_{k+2} (instead of all $b_k, \ldots, b_\ell$), then ID can only issue private keys for three generations of descendants, and those descendants' private keys will be limited even further.

4.2 HIBE with Short Private Keys

Certain applications, such as the time lock encryption (to be described in Section 5), are better served by using a HIBE system with short private keys rather than ciphertexts. We show how to construct a HIBE system whose private key size grows only sublinearly with hierarchy depth.

The idea is to construct a hybrid of the HIBE in Section 3 and the Boneh-Boyen HIBE [1]. Recall that in the former system the private key shrinks as the identity depth increases, while in the latter system the private key grows with the depth of an identity. The hybrid is based on the algebraic similarities between both systems, and exploits their opposite behavior with regard to private key size, to ensure that no private key ever contains more than $O(\sqrt{\ell})$ group elements.

Specifically, for $\omega \in [0,1]$, the hybrid scheme achieves $O(\ell^{\omega}+\ell^{1-\omega})$ private key size and $O(\ell^{\omega})$ ciphertext size at every level in a hierarchy of depth ℓ. The setting $\omega = 0$ corresponds to our HIBE, whereas $\omega = 1$ corresponds to the Boneh-Boyen HIBE [1]. The most efficient hybrids are obtained when $\omega \in [0, 1/2]$. For example, when $\omega = 1/2$, private keys and ciphertexts are of size $O(\sqrt{\ell})$.

Hybrid Scheme. As before, we assume a bilinear group $\mathbb{G}$ and a map $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_1$, where $\mathbb{G}$ and $\mathbb{G}_1$ have prime order p. Let $\ell_1 = \lceil \ell^{\omega} \rceil$ and $\ell_2 = \lceil \ell^{1-\omega} \rceil$. The basic idea is to partition levels of the hierarchy into ℓ_1 consecutive groups of size ℓ_2. Within each group we use the system of Section 3. Between groups we use the Boneh-Boyen HIBE [1].

Let $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ be an identity of depth $k \le \ell$. We will represent ID as a pair (k, I) where $\mathrm{I} \in (\mathbb{Z}_p^*)^{\ell_1 \times \ell_2}$ is an $\ell_1 \times \ell_2$ matrix filled using the elements $\mathrm{I}_1, \ldots, \mathrm{I}_k$ in typographic order: one row at a time starting from the top, in each row starting from the left (note that $\ell_1 \cdot \ell_2 \ge \ell \ge k$; the unfilled matrix entries are undefined). For convenience, we decompose the indices $k = 1, \ldots, \ell$ into row-column pairs (k_1, k_2) such that $k = \ell_2 \cdot (k_1 - 1) + k_2$ where $k_1, k_2 > 0$. For shorthand, we write $(k_1, k_2) = k$. It follows that in the above matrix representation of ID we have $\mathrm{I}_{(i_1,i_2)} = \mathrm{I}_i$ for all $i = 1, \ldots, k$. Or, pictorially, for an ID at the maximum depth ℓ with $\mathrm{I} = \mathrm{I}_1, \ldots, \mathrm{I}_\ell$ and $\ell = \ell_1\ell_2$:

$$\mathrm{I} \;=\; \begin{pmatrix} \mathrm{I}_1 & \mathrm{I}_2 & \ldots & \mathrm{I}_{\ell_2} \\ \mathrm{I}_{\ell_2+1} & \mathrm{I}_{\ell_2+2} & \ldots & \mathrm{I}_{2\,\ell_2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{I}_{(\ell_1-1)\ell_2+1} & \mathrm{I}_{(\ell_1-1)\ell_2+2} & \ldots & \mathrm{I}_{\ell_1\ell_2} \end{pmatrix} \;=\; \begin{pmatrix} \mathrm{I}_{(1,1)} & \mathrm{I}_{(1,2)} & \ldots & \mathrm{I}_{(1,\ell_2)} \\ \mathrm{I}_{(2,1)} & \mathrm{I}_{(2,2)} & \ldots & \mathrm{I}_{(2,\ell_2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{I}_{(\ell_1,1)} & \mathrm{I}_{(\ell_1,2)} & \ldots & \mathrm{I}_{(\ell_1,\ell_2)} \end{pmatrix}.$$

Using this convention, we can now describe the hybrid HIBE system as follows.

$\mathtt{Setup}(\ell, \omega)$: For a HIBE of maximum depth ℓ, first determine ℓ_1 and ℓ_2 as above so that $\ell \le \ell_1 \cdot \ell_2$. Next, select a random generator g in $\mathbb{G}$, a random $\alpha \in \mathbb{Z}_p$, and set $g_1 = g^{\alpha}$. Then, pick random elements $g_2, f_1, \ldots, f_{\ell_1}, h_1, \ldots, h_{\ell_2} \in \mathbb{G}$. The public parameters *params* and the secret *master-key* are given by

$$\textit{params} = (\; g,\; g_1, g_2,\; f_1, \ldots, f_{\ell_1},\; h_1, \ldots, h_{\ell_2} \;), \qquad \textit{master-key} = g_2^{\alpha}.$$

$\mathtt{KeyGen}(d_{\mathsf{ID}|k-1}, \mathsf{ID})$: To generate private key d_{ID} for identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ of depth $(k_1, k_2) = k \le \ell$, where $k_1 \le \ell_1$ and $k_2 \le \ell_2$, pick random $r_1, \ldots, r_{k_1} \in \mathbb{Z}_p$, and output

$$\begin{aligned} d_{\mathsf{ID}} \;=\; \Bigg(& g_2^{\alpha} \cdot \Bigg(\prod_{i=1}^{k_1-1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i \right)^{r_i} \Bigg) \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1} \right)^{r'_{k_1}}, \\ & g^{r_1},\; \ldots,\; g^{r_{k_1-1}},\; g^{r'_{k_1}},\; h_{k_2+1}^{r'_{k_1}},\; \ldots,\; h_{\ell_2}^{r'_{k_1}} \Bigg) \;\in\; \mathbb{G}^{1+k_1+\ell_2-k_2}. \end{aligned} \tag{3}$$

Note that the factors $(\ldots)^{r_i}$ under the $\prod$ sign contain ℓ_2 identity terms each, whereas the last factor $(\ldots)^{r_{k_1}}$ only has k_2 such terms. The size of d_{ID} grows with

k_1 and shrinks with k_2; the private key thus becomes alternatively shorter and longer as the depth of ID increases, but never exceeds $\ell_1 + \ell_2$ elements of $\mathbb{G}$.

The private key for ID can be generated with a private key for $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1}) \in (\mathbb{Z}_p^*)^{k-1}$ as required. Decompose k as (k_1, k_2) according to our convention. There are two cases:

1. If $k-1$ is written $(k_1, k_2 - 1)$, namely k and $k-1$ have the same row index k_1, then we know that the private key for $\mathsf{ID}_{|k-1}$ is of the form:

$$d_{\mathsf{ID}|k-1} = \Bigg(g_2^{\alpha} \cdot \prod_{i=1}^{k_1-1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i\right)^{r_i} \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2-1}^{\mathrm{I}_{(k_1,k_2-1)}} \cdot f_{k_1}\right)^{r_{k_1}}, \;\; g^{r_1}, \;\; \ldots, \;\; g^{r_{k_1}}, h_{k_2}^{r_{k_1}}, \;\; \ldots, \;\; h_{\ell_2}^{r_{k_1}} \Bigg) \;=\; (a_0, b_1, \ldots, b_{k_1}, c_{k_2}, \ldots, c_{\ell_2}) \;\; \in \mathbb{G}^{2+k_1+\ell_2-k_2}.$$

In this case, to generate d_{ID} from $d_{\mathsf{ID}|k-1}$, pick a random $r^* \in \mathbb{Z}_p$ and output

$$d_{\mathsf{ID}} = \Bigg(a_0 \cdot c_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\right)^{r^*}, \;\; b_1, \;\; \ldots, \;\; b_{k_1-1}, \;\; b_{k_1} \cdot g^{r^*}, \;\; c_{k_2+1} \cdot h_{k_2+1}^{r^*}, \;\; \ldots, \;\; c_{\ell_2} \cdot h_{\ell_2}^{r^*} \Bigg) \;\; \in \mathbb{G}^{1+k_1+\ell_2-k_2}.$$

This tuple is of the same form as Eq (3) where $r'_{k_1} = r_{k_1} + r^*$.

2. If the row indices differ, then necessarily $k - 1 = (k_1 - 1, \ell_2)$ and $k = (k_1, 1)$, and the private key for $\mathsf{ID}_{|k-1}$ must be of the form:

$$\begin{aligned} d_{\mathsf{ID}|k-1} &= \Bigg(g_2^{\alpha} \cdot \prod_{i=1}^{k_1-1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i\right)^{r_i}, \;\; g^{r_1}, \;\; \ldots, \;\; g^{r_{k_1-1}} \Bigg) \\ &= (a_0, b_1, \ldots, b_{k_1-1}) \;\; \in \mathbb{G}^{k_1}. \end{aligned}$$

In this case, to generate d_{ID} from $d_{\mathsf{ID}|k-1}$, pick a random $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{ID}} = \Bigg(a_0 \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdot f_{k_1}\right)^{r}, \;\; b_1, \;\; \ldots, \;\; b_{k_1-1}, \;\; g^r, \;\; h_2^r, \;\; , \ldots, h_{\ell_2}^r \Bigg) \;\; \in \mathbb{G}^{k_1+\ell_2}.$$

Again, this tuple conforms to Eq (3) in which r_{k_1} has been set to r.

`Encrypt(`***params*****, ID, M):** To encrypt a message $M \in \mathbb{G}_1$ under the public key $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in \mathbb{Z}_p^k$ where $k = (k_1, k_2)$, pick a random $s \in \mathbb{Z}_p$ and output

$$\mathsf{CT} = \Big(e(g_1, g_2)^s \cdot M, \;\; g^s, \;\; \left(h_1^{\mathrm{I}_{(1,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(1,\ell_2)}} \cdot f_1\right)^s, \;\; \ldots \;\; , \left(h_1^{\mathrm{I}_{(k_1-1,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(k_1-1,\ell_2)}} \cdot f_{k_1-1}\right)^s, \;\; \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\right)^s \Big) \;\; \in \mathbb{G}_1 \times \mathbb{G}^{1+k_1}.$$

`Decrypt(`d_{ID}**, CT):** Consider an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ with $k = (k_1, k_2)$. To decrypt a ciphertext $\mathsf{CT} = (A, B, C_1, \ldots, C_{k_1-1}, C_{k_1})$ using the private key $d_{\mathsf{ID}} = (a_0, b_1, \ldots, b_{k_1}, c_{k_2+1}, \ldots, c_{\ell_2})$, output

$$A \cdot \prod_{i=1}^{k_1} e(b_i, C_i) \Big/ e(B, a_0) = M.$$

Note that the private key components $c_{k_2+1}, \ldots, c_{\ell_2}$ are not used for decryption.

Complexity. It is easy to see that in a hierarchy of depth ℓ, private keys at all levels contain at most $\ell_1 + \ell_2$ elements of $\mathbb{G}$, while ciphertexts contain at most $1 + \ell_1$ elements of $\mathbb{G}$ and one element of $\mathbb{G}_1$. Encryption, decryption, and one-level-down key generation, all require $O(\ell_1 + \ell_2)$ operations, or $O(\sqrt{\ell})$ for the choice $\omega = 1/2$ as claimed. We note that the combination of having a selectable parameter ω together with the option of using an asymmetric bilinear group geared toward reducing the ciphertext or the private key size (described in Section 4.3), gives great flexibility toward achieving the optimal trade-off for a given application.

Security. We prove security based on the $(\ell_2 + 1)$-BDHE assumption (observe that the BDHE assumption implies the BDH assumption). We note that for $\omega = 1/2$, security for a ℓ-level hierarchy is based on the $O(\sqrt{\ell})$-BDHE assumption.

Theorem 2. *Let $\mathbb{G}$ be a bilinear group of prime order p. Consider a hybrid ℓ-HIBE system with identity hierarchy partitioned into ℓ_1 groups each of size ℓ_2. Suppose the decision $(t, \epsilon, \ell_2 + 1)$-BDHE assumption holds in $\mathbb{G}$. Then the hybrid ℓ-HIBE system is (t', q_S, ϵ)-selective identity, chosen plaintext* (IND-sID-CPA) *secure for arbitrary ℓ, q_S, and $t' < t - \Theta(\tau\, \ell\, q_S)$, where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

The proof is similar to that for Theorem 1 and is in the full paper [3].

4.3 Asymmetric Bilinear Groups and MNT Curves

It is often desirable to use bilinear maps $e : \mathbb{G} \times \mathbb{G}' \to \mathbb{G}_1$ where $\mathbb{G}$ and $\mathbb{G}'$ are distinct groups. Such maps let us take advantage of certain curves called MNT curves [20]. Typically, elements of the group $\mathbb{G}$ tend to afford a particularly compact representation compared to elements of $\mathbb{G}'$. This property is used for constructing short signatures [8, 2, 4]. For our system, we can use this property to shrink either the private keys or the ciphertexts. Details are in the full paper.

5 Applications

We now discuss applications of our compact HIBE system and its extensions.

5.1 Forward Secure Encryption

The main purpose of a forward secure encryption scheme is to guarantee that all messages encrypted before the secret key is compromised remain secret.

An elegant public key encryption scheme with forward security was proposed by Canetti, Halevi, and Katz (CHK) [9]. Let $T = 2^t$ be the number of distinct

time periods in the forward secure system. When implemented with previous HIBE systems [14, 1], the CHK framework results in ciphertexts of size $O(t)$ and private keys of size $O(t^2)$. Using public updateable storage, Canetti et al. reduce private key size to $O(t)$ without affecting ciphertext length — the idea is to encrypt the private key for time period i under the public key of time period $i-1$ and store the resulting ciphertext, of size $O(t^2)$, in public storage; consequently, only one HIBE private key of size $O(t)$ is kept in private storage.

Using the HIBE system of Section 3 in the CHK framework, we obtain a forward secure encryption scheme with $O(1)$ ciphertext size and decryption time — independent of the number of time periods. Private keys using our basic system are of size $O(t^2)$. Alternatively, using the hybrid HIBE of Section 4.2 in which we set $\omega = 1/2$, we obtain a forward secure encryption scheme with private key size $O(t^{3/2})$; in this case ciphertext size and decryption time become $O(\sqrt{t})$.

Following Canetti et al. [9], we can store most of the private key in updateable public storage in order to lessen the private storage requirement. Applied to our basic forward secure system, using $O(t^2)$ public storage we can reduce the private key size to $O(t)$ while keeping the ciphertext size constant. Using the hybrid HIBE system (for $\omega = 1/2$), the private storage requirement can be similarly reduced to $O(\sqrt{t})$ at the cost of $O(t^{3/2})$ updateable public storage; ciphertext size in this case remains $O(\sqrt{t})$.

5.2 Forward Secure HIBE

Recently, a forward secure HIBE scheme was proposed by Yao et al. [27]. Their scheme essentially uses two HIBE hierarchies in the manner of Canetti et al. [9] to obtain forward security together with the ability to derive subordinate keys. Their system has ciphertexts of size $O(\ell \cdot t)$ where ℓ is the depth of the identity hierarchy and $T = 2^t$ is the number of time periods. Indeed, they pose as an open problem if a forward secure HIBE scheme with "linear" complexity is possible.

Instantiating both hierarchies in their construction with our HIBE system immediately gives a forward secure HIBE scheme with ciphertexts of size $O(1)$, which resolves this question.

We also propose a more specific forward secure HIBE construction that achieves "linear" $O(\ell + t)$ size for all components, including private keys and public parameters (ciphertexts are no longer constant size in that construction). The construction is a hybrid between the HIBE given in Section 3 and the Boneh-Boyen HIBE from [1]; it is described in detail in the full paper [3].

5.3 Public Key NNL Broadcast Encryption

Broadcast encryption schemes, introduced by Fiat and Naor [13], are cryptosystems designed for the efficient broadcast of data to a dynamic group of users authorized to receive the data. Naor, Naor, and Lotspiech [22] considered broadcast encryption in the stateless receiver setting; they provided a general "subset cover" framework for such broadcast encryption schemes and gave two instances of the framework — the Complete Subtree (CS) method and the more efficient

Subset Difference (SD) method. Further improvements have been proposed such as the Layered Subset Difference (LSD) [16] and the Stratified Subset Difference (SSD) [15]. In the symmetric key setting, only a "center" that possesses the secret keys can broadcast to the users. In a public key broadcast encryption system, anyone is allowed to broadcast to selected subsets of users.

Using the HIBE framework, Dodis and Fazio [11] showed how to translate the SD and LSD methods to the public key setting. For N users and r revoked users, their SD and LSD constructions based on previous HIBE systems give ciphertexts of size $O(r \cdot \log N)$, which is no better than the basic CS method. Substituting the HIBE system of Section 3 restores the full benefits of both SD and LSD, which results in ciphertexts of size $O(r)$.

5.4 Encrypting to the Future

Mont et al. [21] observed that an IBE system gives a mechanism for encrypting to the future using a trusted server. Let D be a certain date string. We view D as a public key in an IBE system. Every day, a trusted server publishes the private key corresponding to that day, which enables messages encrypted for that day to be decrypted. Methods for encrypting to the future without a trusted server were proposed by Rivest, Shamir, and Wagner [23].

One problem with the IBE timelock mechanism is that after n days have passed, the server has to publish a bulletin board with n private keys on it (one private key for each day). The amount of data on the bulletin board can be greatly reduced by using the CHK forward secure encryption scheme *in reverse*. Suppose the CHK framework is set up for a total of T time periods (using a tree of depth $\log_2 T$). To encrypt a message for day $n < T$, use the CHK public key for time period $T - n$. Similarly, on day n the trusted server publishes the CHK private key corresponding to time period $T - n$. This single private key enables anyone to derive the private keys for CHK time periods $T - n, T - n + 1, \ldots, T$. Anyone can thus decrypt messages intended for days in the range $1, \ldots, n$.

Implementing this encoding using our $O(1)$ ciphertext HIBE, the trusted server on any day only needs to publish a single private key comprising $O(\log^2 T)$ group elements. Using the hybrid HIBE system of Section 4.2, the private key posted by the server is further reduced to $O(\log^{3/2} T)$ group elements for ciphertexts of size $O(\sqrt{\log T})$. These parameters are much better than the IBE based mechanism [21], where the bulletin board contains as many as T group elements.

6 Conclusions and Open Problems

We presented a new HIBE system where the ciphertexts consist of three group elements and decryption only requires computing two bilinear maps, both of which are independent of the hierarchy depth. Encryption time is as efficient as other HIBE systems. For a hierarchy of depth ℓ we proved security based on the $(\ell+1)$-BDHE assumption. We expect ℓ-BDHE to be very useful for constructing cryptosystems with short ciphertexts. For example, ℓ-BDHE was recently used to

construct a broadcast encryption system [6] where both ciphertexts and private keys are short.

We discussed several applications of our system, including efficient forward secure encryption, an efficient public key version of the NNL broadcast encryption system, and an efficient mechanism for encrypting to the future. Our HIBE system allows for limited delegation and can be combined with the Boneh-Boyen HIBE to form a hybrid HIBE that has sublinear private key size.

We note that our selective-ID proof of security is tight. On the other hand, the proof of full security (either in the random oracle or standard model) discussed in Section 3.2 degrades exponentially in the hierarchy depth. The same is true for all existing HIBE systems. It is an open problem to construct a HIBE system where security does not degrade exponentially in the hierarchy depth.

Acknowledgments

The authors thank Mihir Bellare for his helpful comments.

References

1. D. Boneh and X. Boyen. Efficient selective-ID identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 223–38. Springer, 2004.
2. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.
3. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/015, 2005.
4. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Proceedings of Crypto 2004*, LNCS, pages 41–55. Springer, 2004.
5. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 213–29. Springer, 2001.
6. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. Cryptology ePrint Archive, Report 2005/018, 2005.
7. D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In *Proceedings of RSA-CT 2005*, 2005.
8. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–32. Springer, 2001.
9. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*. Springer, 2003.
10. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–22. Springer, 2004.

11. Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In J. Feigenbaum, editor, *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.
12. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *Proceedings of the Workshop on Theory and Practice in Public Key Cryptography 2005*, 2005.
13. A. Fiat and M. Naor. Broadcast encryption. In D. Stinson, editor, *Proceedings of Crypto 1993*, volume 773 of *LNCS*, pages 480–91. Springer, 1993.
14. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *Proceedings of Asiacrypt 2002*, volume 2501 of *LNCS*, pages 548–66, 2002.
15. M. Goodrich, J. Sun, and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 511–27. Springer, 2004.
16. D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 47–60, 2002.
17. J. Horwitz and B. Lynn. Towards hierarchical identity-based encryption. In L. Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 466–81. Springer, 2002.
18. A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of Algorithmic Number Theory Symposium IV*, volume 1838 of *LNCS*, pages 385–94. Springer, 2000.
19. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Transactions Fundamentals*, E85-A(2):481–84, 2002.
20. A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–43, 2001.
21. M. C. Mont, K. Harrison, and M. Sadler. The HP time vault service: exploiting IBE for timed release of confidential information. In *Proceedings of the International World Wide Web Conference 2003*, pages 160–69. ACM, 2003.
22. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 41–62. Springer, 2001.
23. R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, MIT Laboratory for Computer Science, 1996.
24. A. Shamir. Identity-based cryptosystems and signature schemes. In G. Blakley and D. Chaum, editors, *Proceedings of Crypto 1984*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.
25. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Proceedings of Eurocrypt 1997*, volume 1233 of *LNCS*, pages 256–66. Springer, 1997.
26. B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, LNCS. Springer, 2005.
27. D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In B. Pfitzmann, editor, *Proceedings of the ACM Conference on Computer and Communications Security 2004*, pages 354–63, 2004.

Fuzzy Identity-Based Encryption

Amit Sahai[1,⋆] and Brent Waters[2]

[1] University of California, Los Angeles
sahai@cs.ucla.edu
[2] Stanford University
bwaters@cs.stanford.edu

Abstract. We introduce a new type of Identity-Based Encryption (IBE) scheme that we call Fuzzy Identity-Based Encryption. In Fuzzy IBE we view an identity as set of descriptive attributes. A Fuzzy IBE scheme allows for a private key for an identity, ω, to decrypt a ciphertext encrypted with an identity, ω', if and only if the identities ω and ω' are close to each other as measured by the "set overlap" distance metric. A Fuzzy IBE scheme can be applied to enable encryption using biometric inputs as identities; the error-tolerance property of a Fuzzy IBE scheme is precisely what allows for the use of biometric identities, which inherently will have some noise each time they are sampled. Additionally, we show that Fuzzy-IBE can be used for a type of application that we term "attribute-based encryption".

In this paper we present two constructions of Fuzzy IBE schemes. Our constructions can be viewed as an Identity-Based Encryption of a message under several attributes that compose a (fuzzy) identity. Our IBE schemes are both error-tolerant and secure against collusion attacks. Additionally, our basic construction does not use random oracles. We prove the security of our schemes under the Selective-ID security model.

1 Introduction

Identity-Based Encryption [15] (IBE) allows for a sender to encrypt a message to an identity without access to a public key certificate. The ability to do public key encryption without certificates has many practical applications. For example, a user can send an encrypted mail to a recipient, e.g. bobsmith@gmail.com, without the requiring either the existence of a Public-Key Infrastructure or that the recipient be on-line at the time of creation.

One common feature of all previous Identity-Based Encryption systems is that they view identities as a string of characters. In this paper we propose a new type of Identity-Based Encryption that we call *Fuzzy Identity-Based Encryption* in which we view identities as a set of descriptive attributes. In a Fuzzy Identity-Based Encryption scheme, a user with the secret key for the identity ω is able

⋆ Amit Sahai's research was supported by generous grant s from the NSF ITR program, as well as a Sloan Foundation Fellowship.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 457–473, 2005.

to decrypt a ciphertext encrypted with the public key ω' if and only if ω and ω' are within a certain distance of each other as judged by some metric. Therefore, our system allows for a certain amount of error-tolerance in the identities.

Fuzzy-IBE gives rise to two interesting new applications. The first is an Identity-Based Encryption system that uses biometric identities. That is we can view a user's biometric, for example an iris scan, as that user's identity described by several attributes and then encrypt to the user using their biometric identity. Since biometric measurements are noisy, we cannot use existing IBE systems. However, the error-tolerance property of Fuzzy-IBE allows for a private key (derived from a measurement of a biometric) to decrypt a ciphertext encrypted with a slightly different measurement of the same biometric.

Secondly, Fuzzy IBE can be used for an application that we call "attribute-based encryption". In this application a party will wish to encrypt a document to all users that have a certain set of attributes. For example, in a computer science department, the chairperson might want to encrypt a document to all of its systems faculty on a hiring committee. In this case it would encrypt to the identity {"hiring-committee","faculty","systems"}. Any user who has an identity that contains all of these attributes could decrypt the document. The advantage to using Fuzzy IBE is that the document can be stored on an simple untrusted storage server instead of relying on trusted server to perform authentication checks before delivering a document.

We further discuss the usefulness of using biometrics in Identity-Based and then discuss our contributions.

Using biometrics in Identity-Based Encryption. In many situations, using biometric-based identity in an IBE system has a number of important advantages over "standard" IBE. We argue that the use of biometric identities fits the framework of Identity-Based Encryption very well and is a very valuable application of it.

First, the process of obtaining a secret key from an authority is very natural and straightforward. In standard Identity-Based Encryption schemes a user with a certain identity, for example, "Bob Smith", will need to go to an authority to obtain the private key corresponding to the identity. In this process the user will need to "prove" to the authority that he is indeed entitled to this identity. This will typically involve presenting supplementary documents or credentials. The type of authentication that is necessary is not always clear and robustness of this process is questionable (the supplementary documents themselves could be subject to forgery). Typically, there will exist a tradeoff between a system that is expensive in this step and one that is less reliable.

In contrast, if a biometric is used as an identity then the verification process for an identity is very clear. The user must demonstrate ownership of the biometric under the supervision of a well trained operator. If the operator is able to detect imitation attacks, for example playing the recording of a voice, then the security of this phase is only limited by the quality of the biometric technique itself. We emphasize that the biometric measurement for an individual need *not* be kept secret. Indeed, it is not if it is used as a public key. We must

only guarantee that an attacker cannot fool the key authority into believing that an attacker owns a biometric identity that he does not.

Also, a biometric identity is an inherent trait and will always with a person. Using biometrics in Identity-Based Encryption will mean that the person will always have their public key handy. In several situations a user will want to present an encryption key to someone when they are physically present. For example, consider the case when a user is traveling and another party encrypts an ad-hoc meeting between them.

Finally, using a biometric as an identity has the advantage that identities are unique if the underlying biometric is of a good quality. Some types of standard identities, such as the name "Bob Smith" will clearly not be unique or change owners over time.

Security Against Collusion Attacks. In addition to providing error-tolerance in the set of attributes composing the identity any IBE scheme that encrypts to multiple attributes must provide security against collusion attacks. In particular, no group of users should be able to combine their keys in such a way that they can decrypt a ciphertext that none of them alone could. This property is important for security in both biometric applications and "attribute-based encryption".

Our Contributions. We formalize the notion of Fuzzy Identity-Based Encryption and provide a construction for a Fuzzy Identity-Based Encryption scheme. Our construction uses groups for which an efficient bilinear map exists, but for which the Computational Diffie-Hellman problem is assumed to be hard.

Our primary technique is that we construct a user's private key as a set of private key components, one for each attribute in the user's identity. We share use Shamir's method of secret sharing [14] to distribute shares of a master secret in the exponents of the user's private key components. Shamir's secret sharing within the exponent gives our scheme the crucial property of being error-tolerant since only a subset of the private key components are needed to decrypt a message. Additionally, our scheme is resistant to collusion attacks. Different users have their private key components generated with different random polynomials. If multiple users collude they will be unable to combine their private key components in any useful way.

In the first version of our scheme, the public key size grows linearly with the number of potential attributes in the universe. The public parameter growth is manageable for a biometric system where all the possible attributes are defined at the system creation time. However, this becomes a limitation in a more general system where we might like an attribute to be defined by an arbitrary string. To accommodate these more general requirements we additionally provide a Fuzzy-IBE system for large universes, where attributes are defined by arbitrary strings.

We prove our scheme secure under an adapted version of the Selective-ID security model first proposed by Canetti et al. [5]. Additionally, our construction does not use random oracles. We reduce the security of our scheme to an assumption that is similar to the Decisional Bilinear Diffie-Hellman assumption.

1.1 Related Work

Identity-Based Encryption. Shamir [15] first proposed the concept of Identity-Based Encryption. However, it wasn't until much later that Boneh and Franklin [3] presented the first Identity-Based Encryption scheme that was both practical and secure. Their solution made novel use of groups for which there was an efficiently computable bilinear map.

Canetti et al. [5] proposed the first construction for IBE that was provably secure outside the random oracle model. To prove security they described a slightly weaker model of security known as the Selective-ID model, in which the adversary declares which identity he will attack before the global public parameters are generated. Boneh and Boyen [2] give two schemes with improved efficiency and prove security in the Selective-ID model without random oracles.

Biometrics. Other work in applying biometrics to cryptography has focused on the derivation of a secret from a biometric [12, 11, 10, 6, 9, 7, 4]. This secret can be then used for operations such as symmetric encryption or UNIX style password authentication.

The distinguishing feature of our work from the above related work on biometrics above is that we view the biometric input as potentially *public* information instead of a secret. Our only physical requirement is that the biometric cannot be imitated such that a trained human operator would be fooled. We stress the importance of this, since it is much easier to capture a digital reading of someone's biometric, than to fool someone into believing that someone else's biometric is one's own. Simply capturing a digital reading of someone's biometric would (forever) invalidate approaches where symmetric keys are systematically derived from biometric readings.

Attribute-based encryption. Yao et al. [17] show how an IBE system that encrypts to multiple hierarchical-identities in a collusion-resistant manner implies a forward secure Hierarchical IBE scheme. They also note how their techniques for resisting collusion attacks are useful in attribute-based encryption. However, the cost of their scheme in terms of computation, private key size, and ciphertext size increases exponentially with the number of attributes.

1.2 Organization

The rest of the paper is organized as follows. In Section 2 we formally define a Fuzzy Identity-Based Encryption scheme including the Selective-ID security model for one. Then, we describe our security assumptions. In Section 3 we show why two naive approaches do not work. We follow with a description of our construction in Section 4 and in Section 5 we prove the security of our scheme. We describe our second construction in Section 6. Finally, we conclude in Section 7.

2 Preliminaries

We begin by presenting our definition of security. We follow with a brief review of bilinear maps, and then state the complexity assumptions we use for our proofs of security.

2.1 Definitions

In this section we define our Selective-ID models of security for Fuzzy Identity Based Encryption. The Fuzzy Selective-ID game is very similar to the standard Selective-ID model for Identity-Based Encryption with the exception that the adversary is only allowed to query for secret keys for identities which have less than d overlap with the target identity.

Fuzzy Selective-ID.

Init. The adversary declares the identity, α, that he wishes to be challenged upon.

Setup. The challenger runs the setup phase of the algorithm and tells the adversary the public parameters.

Phase 1. The adversary is allowed to issue queries for private keys for many identities, γ_j, where $|\gamma_j \cap \alpha| < d$ for all j.

Challenge. The adversary submits two equal length messages M_0, M_1. The challenger flips a random coin, b, and encrypts M_b with α. The ciphertext is passed to the adversary.

Phase 2. Phase 1 is repeated.

Guess. The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $\Pr[b' = b] - \frac{1}{2}$.

Definition 1 (Fuzzy Selective-ID). *A scheme is secure in the Fuzzy Selective-ID model of security if all polynomial-time adversaries have at most a negligible advantage in the above game.*

2.2 Bilinear Maps

We briefly review the facts about groups with efficiently computable bilinear maps. We refer the reader to previous literature [3] for more details.

Let $\mathbb{G}_1, \mathbb{G}_2$ be groups of prime order p, and let g be a generator of $\mathbb{G}_1$. We say $\mathbb{G}_1$ has an admissible bilinear map, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, into $\mathbb{G}_2$ if the following two conditions hold. The map is bilinear; for all a, b we have $e(g^a, g^b) = e(g, g)^{ab}$. The map is non-degenerate; we must have that $e(g, g) \neq 1$.

2.3 Complexity Assumptions

We state our complexity assumptions below.

Definition 2 (Decisional Bilinear Diffie-Hellman (BDH) Assumption). *Suppose a challenger chooses $a, b, c, z \in \mathbb{Z}_p$ at random. The Decisional BDH*

assumption is that no polynomial-time adversary is to be able to distinguish the tuple $(A = g^a, B = g^b, C = g^c, Z = e(g,g)^{abc})$ *from the tuple* $(A = g^a, B = g^b, C = g^c, Z = e(g,g)^z)$ *with more than a negligible advantage.*

Definition 3 (Decisional Modified Bilinear Diffie-Hellman (MBDH) Assumption). *Suppose a challenger chooses* $a, b, c, z \in \mathbb{Z}_p$ *at random. The Decisional MBDH assumption is that no polynomial-time adversary is to be able to distinguish the tuple* $(A = g^a, B = g^b, C = g^c, Z = e(g,g)^{\frac{ab}{c}})$ *from* $(A = g^a, B = g^b, C = g^c, Z = e(g,g)^z)$ *with more than a negligible advantage.*

3 Other Approaches

Before describing our scheme we first show three potential approaches to building a Fuzzy Identity-Based Encryption scheme and show why they fall short. This discussion additionally motivates our approach to the problem.

Correcting the error. We consider the feasibility of "correcting" the errors of a biometric measurement and then use standard Identity-Based Encryption to encrypt a message under the corrected input. However, this approach relies upon the faulty assumption that each biometric input measurement is slightly deviated from some "true" value and that the set of possible "true" values are well known. In practice, the only reasonable assumption is that two measurements sampled from the same person will be within a certain distance of each other. This intuition is captured by previous work. Dodis, Rezyin, and Smith [7] use what they call a *fuzzy sketch* that contains information of a first sampling of a biometric which allows subsequent measurements to be corrected to it. If the correction could be done without any additional information then we could simply do away with the fuzzy sketch.

Key per Attribute. The second naive approach we consider is for an authority to give a user a different private key for each of the attributes that describe the user. Such a system easily falls prey to simple collusion attacks where multiple users combine their keys to form identities that are a combination of their attributes. The colluders are then able to decrypt ciphertexts that none of them individually were able to decrypt.

Several Keys. Suppose a key authority measures an input ω for a particular party. The authority could create a separate standard IBE private key for every ω' such that $|\omega \cap \omega'| \geq d$, for some error-tolerance parameter d. However, the private key storage will grow exponentially in d and the system will be impractical for even modest values of d.

4 Our Construction

Recall that we view identities as sets of attributes and we let the value d represent the error-tolerance in terms of minimal set overlap. When an authority is creating

a private key for a user he will associate a random $d-1$ degree polynomial, $q(x)$, with each user with the restriction that each polynomial have the same valuation at point 0, that is $q(0) = y$.

For each of the attributes associated with a user's identity the key generation algorithm will issue a private key component that is tied to the user's random polynomial $q(x)$. If the user is able to "match" at least d components of the ciphertext with their private key components, then they will be able to perform decryption. However, since the private key components are tied to random polynomials, multiple user's are unable to combine them in anyway that allows for collusion attacks.

A detailed description of our scheme follows.

4.1 Description

Recall that we wish to create an IBE scheme in which a ciphertext created using identity ω can be decrypted only by a secret key ω' where $|\omega \cap \omega'| \geq d$.

Let $\mathbb{G}_1$ be bilinear group of prime order p, and let g be a generator of $\mathbb{G}_1$. Additionally, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote the bilinear map. A security parameter, κ, will determine the size of the groups.

We also define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$:

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}.$$

Identities will be element subsets of some universe, $\mathcal{U}$, of size $|\mathcal{U}|$. We will associate each element with a unique integer in ${\mathbb{Z}_p}^*$. (In practice an attribute will be associated with each element so that identities will have some semantics.) Our construction follows:

Setup(d). First, define the universe, $\mathcal{U}$ of elements. For simplicity, we can take the first $|\mathcal{U}|$ elements of ${\mathbb{Z}_p}^*$ to be the universe. Namely, the integers $1, \ldots, |\mathcal{U}| \pmod p$.

Next, choose $t_1, \ldots, t_{|\mathcal{U}|}$ uniformly at random from $\mathbb{Z}_p$. Finally, choose y uniformly at random in $\mathbb{Z}_p$. The published public parameters are:

$$T_1 = g^{t_1}, \ldots, T_{|\mathcal{U}|} = g^{t_{|\mathcal{U}|}}, Y = e(g,g)^y.$$

The master key is:

$$t_1, \ldots, t_{|\mathcal{U}|}, y.$$

Key Generation. To generate a private key for identity $\omega \subseteq \mathcal{U}$ the following steps are taken. A $d-1$ degree polynomial q is randomly chosen such that $q(0) = y$. The private key consists of components, $(D_i)_{i \in \omega}$, where $D_i = g^{\frac{q(i)}{t_i}}$ for every $i \in \omega$.

Encryption. Encryption with the public key ω' and message $M \in \mathbb{G}_2$ proceeds as follows.

First, a random value $s \in \mathbb{Z}_p$ is chosen. The ciphertext is then published as:

$$E = (\omega', E' = MY^s, \{E_i = T_i^s\}_{i \in \omega'}).$$

Note that the identity, ω', is included in the ciphertext.

Decryption. Suppose that a ciphertext, E, is encrypted with a key for identity ω' and we have a private key for identity ω, where $|\omega \cap \omega'| \geq d$. Choose an arbitrary d-element subset, S, of $\omega \cap \omega'$.

Then, the ciphertext can be decrypted as:

$$\begin{aligned} & E' / \prod_{i \in S} (e(D_i, E_i))^{\Delta_{i,S}(0)} \\ &= Me(g,g)^{sy} / \prod_{i \in S} \left(e(g^{\frac{q(i)}{t_i}}, g^{st_i}) \right)^{\Delta_{i,S}(0)} \\ &= Me(g,g)^{sy} / \prod_{i \in S} \left(e(g,g)^{sq(i)} \right)^{\Delta_{i,S}(0)} \\ &= M. \end{aligned}$$

The last equality is derived from using polynomial interpolation in the exponents. Since, the polynomial $sq(x)$ is of degree $d-1$ it can be interpolated using d points.

4.2 Efficiency and Key Sizes

The number of exponentiations in the group $\mathbb{G}_1$ to encrypt to an identity will be linear in the number of elements in the identity's description. The cost of decryption will be dominated by d bilinear map computations.

The number of group elements in the public parameters grows linearly with the number attributes in the system (elements in the defined universe). The number of group elements that compose a user's private key grow linearly with the number of attributes associated with her identity. Finally, the number of group elements in a ciphertext grows linearly with the size of the identity we are encrypting to.

4.3 Flexible Error-Tolerance

In this construction the error-tolerance is set to a fixed value d. However, in practice a party constructing a ciphertext might want more flexibility. For example, if a biometric input device happens to be less reliable it might be desirable to relax the set overlap parameters. In the example of attribute-based encryption we would like to have flexibility in the number of attributes required to access a document.

There are two simple methods for achieving flexible error-tolerance. First, we can create multiple systems with different values of d and the party encrypting a message can choose the appropriate one. For m different systems the size of the public parameters and private keys both increase by a factor of m. In the second method the authority will reserve some attributes that it will issue to every key-holder as part of their identity. The party encrypting the message can increase the error-tolerance by increasing the number of these "default" attributes it includes in the encryption identity. In this approach ciphertexts must be at least as long as the maximum number of attributes that can be required in an encryption. Additionally, we can combine the above two techniques and explore tradeoffs between ciphertext size and public parameter and private key size.

5 Proof of Security

We prove that the security of our scheme in the Selective-ID model reduces to the hardness of the Decisional MBDH assumption.

Theorem 1. *If an adversary can break our scheme in the Fuzzy Selective ID Model, then a simulator can be constructed to play the Decisional MBDH game with a non-negligible advantage.*

Proof. Suppose there exists a polynomial-time adversary, $\mathcal{A}$, that can attack our scheme in the Selective-ID model with advantage ϵ. We build a simulator $\mathcal{B}$ that can play the Decisional MBDH game with advantage $\frac{\epsilon}{2}$. The simulation proceeds as follows:

We first let the challenger set the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with an efficient bilinear map, e and generator g. The challenger flips a fair binary coin, μ, outside of $\mathcal{B}$'s view. If $\mu = 0$, the challenger sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^{\frac{ab}{c}})$; otherwise it sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^z)$ for random a, b, c, z. We assume the universe, $\mathcal{U}$ is defined.

Init. The simulator $\mathcal{B}$ runs $\mathcal{A}$ and receives the challenge identity, α.

Setup. The simulator assigns the public key parameters as follows. It sets the parameter $Y = e(g, A) = e(g, g)^a$. For all $i \in \alpha$ it chooses random $\beta_i \in \mathbb{Z}_p$ and sets $T_i = C^{\beta_i} = g^{c\beta_i}$. For all $i \in \mathcal{U} - \alpha$ it chooses random $w_i \in \mathbb{Z}_p$ and sets $T_i = g^{w_i}$.

It then gives the public parameters to $\mathcal{A}$. Notice that from the view $\mathcal{A}$ all parameters are chosen at random as in the construction.

Phase 1. $\mathcal{A}$ makes requests for private keys where the identity set overlap between the identities for each requested key and α is less than d.

Suppose $\mathcal{A}$ requests a private key γ where $|\gamma \cap \alpha| < d$. We first define three sets Γ, Γ', S in the following manner:

$$\Gamma = \gamma \cap \alpha,$$

$$\Gamma' \text{ be any set such that } \Gamma \subseteq \Gamma' \subseteq \gamma \text{ and } |\Gamma'| = d-1, \text{and}$$
$$S = \Gamma' \cup \{0\}.$$

Next, we define the decryption key components, D_i, for $i \in \Gamma'$ as:

$$\text{If } i \in \Gamma: \ D_i = g^{s_i} \text{ where } s_i \text{ is chosen randomly in } \mathbb{Z}_p.$$
$$\text{If } i \in \Gamma' - \Gamma: \ D_i = g^{\frac{\lambda_i}{w_i}} \text{ where } \lambda_i \text{ is chosen randomly in } \mathbb{Z}_p.$$

The intuition behind these assignments is that we are implicitly choosing a random $d-1$ degree polynomial $q(x)$ by choosing its value for the $d-1$ points randomly in addition to having $q(0) = a$. For $i \in \Gamma$ we have $q(i) = c\beta_i s_i$ and for $i \in \Gamma' - \Gamma$ we have $q(i) = \lambda_i$.

The simulator can calculate the other D_i values where $i \notin \Gamma'$ since the simulator knows the discrete log of T_i for all $i \notin \alpha$. The simulator makes the assignments as follows:

$$\text{If } i \notin \Gamma': \ D_i = \left(\prod_{j \in \Gamma} C^{\frac{\beta_j s_j \Delta_{j,S}(i)}{w_i}}\right)\left(\prod_{j \in \Gamma' - \Gamma} g^{\frac{\lambda_j \Delta_{j,S}(i)}{w_i}}\right) Y^{\frac{\Delta_{0,S}(i)}{w_i}}$$

Using interpolation the simulator is able to calculate $D_i = g^{\frac{q(i)}{t_i}}$ for $i \notin \Gamma'$ where $q(x)$ was implicitly defined by the random assignment of the other $d-1$ variables $D_i \in \Gamma'$ and the variable Y.

Therefore, the simulator is able to construct a private key for the identity γ. Furthermore, the distribution of the private key for γ is identical to that of the original scheme.

Challenge. The adversary, $\mathcal{A}$, will submit two challenge messages M_1 and M_0 to the simulator. The simulator flips a fair binary coin, ν, and returns an encryption of M_ν. The ciphertext is output as:

$$E = (\alpha, E' = M_\nu Z, \{E_i = B^{\beta_i}\}_{i \in \alpha}).$$

If $\mu = 0$, then $Z = e(g,g)^{\frac{ab}{c}}$. If we let $r' = \frac{b}{c}$, then we have $E_0 = M_\nu Z = M_\nu e(g,g)^{\frac{ab}{c}} = M_\nu e(g,g)^{ar'} = M_\nu Y^{r'}$ and $E_i = B^{\beta_i} = g^{b\beta_i} = g^{\frac{b}{c} c\beta_i} = g^{r' c\beta_i} = (T_i)^{r'}$. Therefore, the ciphertext is a random encryption of the message m_ν under the public key α.

Otherwise, if $\mu = 1$, then $Z = g^z$. We then have $E' = M_\nu e(g,g)^z$. Since z is random, E' will be a random element of $\mathbb{G}_2$ from the adversaries view and the message contains no information about M_ν.

Phase 2. The simulator acts exactly as it did in Phase 1.

Guess. $\mathcal{A}$ will submit a guess ν' of ν. If $\nu = \nu'$ the simulator will output $\mu' = 0$ to indicate that it was given a MBDH-tuple otherwise it will output $\mu' = 1$ to indicate it was given a random 4-tuple.

As shown in the construction the simulator's generation of public parameters and private keys is identical to that of the actual scheme.

In the case where $\mu = 1$ the adversary gains no information about ν. Therefore, we have $\Pr[\nu \neq \nu' | \mu = 1] = \frac{1}{2}$. Since the simulator guesses $\mu' = 1$ when $\nu \neq \nu'$, we have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$.

If $\mu = 0$ then the adversary sees an encryption of m_ν. The adversary's advantage in this situation is ϵ by definition. Therefore, we have $\Pr[\nu = \nu' | \mu = 0] = \frac{1}{2} + \epsilon$. Since the simulator guesses $\mu' = 0$ when $\nu = \nu'$, we have $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon$.

The overall advantage of the simulator in the Decisional MBDH game is $\frac{1}{2}\Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2}(\frac{1}{2} + \epsilon) + \frac{1}{2}\frac{1}{2} - \frac{1}{2} = \frac{1}{2}\epsilon$. □

5.1 Chosen-Ciphertext Security

Our security definitions and proofs have been in the chosen-plaintext model. Our scheme can be extended to the chosen-ciphertext model by applying the technique of using simulation-sound NIZK proofs to achieve chosen-ciphertext security [13]. Alternatively, if we are willing to use random oracles, then the we can use standard techniques such as the Fujisaki-Okamoto transformation [8].

5.2 Security in Full IBE Model

Suppose all identities are composed of n attributes and we have a universe of attributes, $\mathcal{U}$. We make the observation [2] that our scheme is secure in the full model with a factor of $\binom{|\mathcal{U}|}{n}$ in the reduction.

The original IBE scheme of Boneh and Franklin [3] and a later schemes of Boneh and Boyen [2] and Waters [16] achieve IBE in the full model with non-exponential reductions. However, all methods achieve this by essentially removing the relationships between nearby identities. In Fuzzy-IBE it is essential that there exists a relationship between nearby identities. Therefore, we conjecture that a scheme that has a non-exponential loss of security in the full model will require significantly different methods than those seen in prior work.

6 Large Universe Construction

In the previous construction the size of the public parameters grows linearly with the number of possible attributes in the universe. We describe a second scheme which uses all elements of ${\mathbb{Z}_p}^*$ as the universe, yet the public parameters only grow linearly in a parameter n, which we fix as the maximum size identity we can encrypt to.

In addition to decreasing the public parameter size, having a large universe allows us to apply a collision-resistant hash function $H : \{0,1\}^* \rightarrow {\mathbb{Z}_p}^*$ and use arbitrary strings as attributes. We can now use attributes that were not necessarily considered during the public key setup. For example, we can add any verifiable attribute, such as "Ran in N.Y. Marathon 2005", to a user's private key.

Our large universe construction is built using similar concepts to the previous scheme and uses an algebraic technique of Boneh and Boyen [2]. Additionally, we reduce the security of this scheme to the Decisional BDH problem. We now describe our construction and give our proof of security.

6.1 Description

Let $\mathbb{G}_1$ be bilinear group of prime order p, and let g be a generator of $\mathbb{G}_1$. Additionally, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote the bilinear map. We restrict encryption identities to be of length n for some fixed n.

We define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$:

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}.$$

Identities will be sets of n elements of ${\mathbb{Z}_p}^*$.[1] Alternatively, we can describe an identity as a collection of n strings of arbitrary length and use a collision resistant hash function, H, to hash strings into members of ${\mathbb{Z}_p}^*$. Our construction follows:

Setup(n, d). First, choose $g_1 = g^y, g_2 \ \in \mathbb{G}_1$.

Next, choose $t_1, \ldots, t_{n+1}$ uniformly at random from $\mathbb{G}_1$. Let N be the set $\{1, \ldots, n+1\}$ and we define a function, T, as:

$$T(x) = g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)}.$$

We can view T as the function $g_2^{x^n} g^{h(x)}$ for some n degree polynomial h. The public key is published as: $g_1, g_2, t_1, \ldots, t_{n+1}$ and the private key is y.

Key Generation. To generate a private key for identity ω the following steps are taken. A $d-1$ degree polynomial q is randomly chosen such that $q(0) = y$. The private key will consist of two sets. The first set, $\{D_i\}_{i \in \omega}$, where the elements are constructed as

$$D_i = g_2^{q(i)} T(i)^{r_i},$$

where r_i is a random member of $\mathbb{Z}_p$ defined for all $i \in \omega$.

The other set is $\{d_i\}_{i \in \omega}$ where the elements are constructed as

$$d_i = g^{r_i}.$$

Encryption. Encryption with the public key ω' and message $M \in \mathbb{G}_2$ proceeds as follows.

First, a random value $s \in \mathbb{Z}_p$ is chosen. The ciphertext is then published as:

$$E = (\omega', E' = M e(g_1, g_2)^s, E'' = g^s, \{E_i = T(i)^s\}_{i \in \omega'}).$$

[1] With some minor modifications to our scheme, which we omit for simplicity, we can encrypt to all identities of size $\leq n$.

Decryption. Suppose that a ciphertext, E, is encrypted with a key for identity ω' and we have a key for identity ω, where $|\omega \cap \omega'| \geq d$. Choose an arbitrary d-element subset, S, of $\omega \cap \omega'$.

Then, the ciphertext can be decrypted as:

$$\begin{aligned} M &= E' \prod_{i \in S} \left(\frac{e(d_i, E_i)}{e(D_i, E'')} \right)^{\Delta_{i,S}(0)} \\ &= Me(g_1, g_2)^s \prod_{i \in S} \left(\frac{e(g^{r_i}, T(i)^s)}{e(g_2^{q(i)} T(i)^{r_i}, g^s)} \right)^{\Delta_{i,S}(0)} \\ &= Me(g_1, g_2)^s \prod_{i \in S} \left(\frac{e(g^{r_i}, T(i)^s)}{e(g_2^{q(i)}, g^s) e(T(i)^{r_i}, g^s)} \right)^{\Delta_{i,S}(0)} \\ &= Me(g, g_2)^{ys} \prod_{i \in S} \frac{1}{e(g, g_2)^{q(i) s \Delta_{i,S}(0)}} \\ &= M. \end{aligned}$$

The last equality is derived from using polynomial interpolation in the exponents. Since, the polynomial $sq(x)$ is of degree $d-1$ it can be interpolated using d points.

6.2 Efficiency and Key Sizes

Again, he number of exponentiations in the group $\mathbb{G}_1$ to encrypt to an identity will be linear in the number of elements in the identity's description. The cost of decryption will be dominated by $2 \cdot d$ bilinear map computations.

The key feature of the scheme is that the number of group elements in the public parameters only grows linearly with, n, the maximum number of attributes that can describe an encryption identity. The number of group elements that compose a user's private key grow linearly with the number of attributes associated with her identity. Finally, the number of group elements in a ciphertext grows linearly with the size of the identity we are encrypting to.

6.3 Proof of Security

We prove that the security of our scheme in the Selective-ID model reduces to the hardness of the Decisional BDH assumption.

Theorem 2. *If an adversary can break our scheme in the Fuzzy Selective ID Model, then a simulator can be constructed to play the Decisional BDH game with a non-negligible advantage.*

Proof. Suppose there exists a polynomial-time adversary, $\mathcal{A}$, that can attack our scheme in the Selective-ID model with advantage ϵ. We build a simulator $\mathcal{B}$ that can play the Decisional BDH game with advantage $\frac{\epsilon}{2}$.

The simulation proceeds as follows:

We first let the challenger set the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with an efficient bilinear map, e and generator g. The challenger flips a fair binary coin μ outside of $\mathcal{B}$'s view. If $\mu = 0$, the challenger sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^{abc})$; otherwise it sets $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^z)$ for random a, b, c, z.

Init. $\mathcal{B}$ will run $\mathcal{A}$ and receive the challenge identity, α, an n element set of members of $\mathbb{Z}_p$.

Setup. The simulator assigns the public parameters $g_1 = A$ and $g_2 = B$. It then chooses a random n degree polynomial $f(x)$ and calculates an n degree polynomial $u(x)$ such that $u(x) = -x^n$ for all $x \in \alpha$ and where $u(x) \neq -x^n$ for some other x. Since $-x^n$ and $u(x)$ are two n degree polynomials they will either agree on at most n points or they are the same polynomial. Our construction assures that $\forall x\ u(x) = -x^n$ if and only if $x \in \alpha$.

Then, for i from 1 to $n+1$ the simulator sets $t_i = g_2^{u(i)} g^{f(i)}$. Note that since $f(x)$ is a random n degree polynomial all t_i will be chosen independently at random as in the construction and we implicitly have $T(x) = g_2^{i^n + u(i)} g^{f(i)}$.

Phase 1. $\mathcal{A}$ makes requests for private keys where the identity set overlap between the identities for the requested keys and α is less than d.

Suppose $\mathcal{A}$ requests a private key γ. We first define three sets Γ, Γ', S in the following manner:

$$\Gamma = \gamma \cap \alpha,$$

$$\Gamma' \text{ be any set such that } \Gamma \subseteq \Gamma' \subseteq \gamma \text{ and } |\Gamma'| = d - 1, \text{and}$$

$$S = \Gamma' \cup \{0\}.$$

Next, we define the decryption key components D_i and d_i for $i \in \Gamma'$ as:

$$D_i = g_2^{\lambda_i} T(i)^{r_i} \text{ where } r_i, \lambda_i \text{ are chosen randomly in } \mathbb{Z}_p \text{ and we let } d_i = g^{r_i}.$$

The intuition behind these assignments is that we are implicitly choosing a random $d-1$ degree polynomial $q(x)$ by choosing its value for the $d-1$ points in Γ randomly by setting $q(i) = \lambda_i$ in addition to having $q(0) = a$.

The simulator also needs to calculate the decryption key values for all $i \in \gamma - \Gamma'$. We calculate these points to be consistent with our implicit choice of $q(x)$. The key components are calculated as:

$$D_i = \left(\prod_{j \in \Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)}\right) \left(g_1^{\frac{-f(i)}{i^n + u(i)}} (g_2^{i^n + u(i)} g^{f(i)})^{r'_i}\right)^{\Delta_{0,S}(i)}$$

and

$$d_i = (g_1^{\frac{-1}{i^n + u(i)}} g^{r'_i})^{\Delta_{0,S}(i)}.$$

The value $i^n + u(i)$ will be non-zero for all $i \notin \alpha$,which includes all $i \in \gamma - \Gamma'$. This follows from the our construction of $u(x)$.

Let $r_i = (r'_i - \frac{a}{i^n+u(i)})\Delta_{0,S}(i)$ and let $q(x)$ be defined as above. We then have:

$$\begin{aligned}
D_i &= (\prod_{j\in\Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)}) \left((g_1^{\frac{-f(i)}{i^n+u(i)}})(g_2^{i^n+u(i)} g^{f(i)})^{r'_i} \right)^{\Delta_{0,S}(i)} \\
&= (\prod_{j\in\Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)}) \left((g^{\frac{-af(i)}{i^n+u(i)}})(g_2^{i^n+u(i)} g^{f(i)})^{r'_i} \right)^{\Delta_{0,S}(i)} \\
&= (\prod_{j\in\Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)}) \left((g_2^a (g_2^{i^n+u(i)} g^{f(i)})^{\frac{-a}{i^n+u(i)}})(g_2^{i^n+u(i)} g^{f(i)})^{r'_i} \right)^{\Delta_{0,S}(i)} \\
&= (\prod_{j\in\Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)}) \left(g_2^a (g_2^{i^n+u(i)} g^{f(i)})^{r'_i - \frac{a}{i^n+u(i)}} \right)^{\Delta_{0,S}(i)} \\
&= (\prod_{j\in\Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)}) g_2^{a\Delta_{0,S}(i)} (T(i))^{r_i} \\
&= g_2^{q(i)} T(i)^{r_i}
\end{aligned}$$

Additionally, we have:

$$d_i = (g_1^{\frac{-1}{i^n+u(i)}} g^{r'_i})^{\Delta_{0,S}(i)} = (g^{r'_i - \frac{a}{i^n+u(i)}})^{\Delta_{0,S}(i)} = g^{r_i}$$

Therefore, the simulator is able to construct a private key for the identity γ. Furthermore, the distribution of the private key for γ is identical to that of the original scheme since our choices of λ_i induce a random $d-1$ degree polynomial and our construction of the private keys components d_i and D_i.

Challenge. The adversary, $\mathcal{A}$, will submit two challenge messages M_1 and M_0 to the simulator. The simulator flips a fair binary coin, ν, and returns an encryption of M_ν. The ciphertext is output as:

$$E = (\alpha, E' = M_\nu Z, E'' = C, \{E_i = C^{f(i)}\}_{i\in\alpha}).$$

If $\mu = 0$, then $Z = e(g,g)^{abc}$. Then the ciphertext is:

$$E = (\alpha, E' = M_\nu e(g,g)^{abc}, E'' = g^c, \{E_i = (g^c)^{f(i)} = T(i)^c\}_{i\in\alpha}).$$

This is a valid ciphertext for the message M_ν under the identity α.

Otherwise, if $\mu = 1$, then $Z = e(g,g)^z$ and $E' = M_\nu e(g,g)^z$. Since z is random, E' will be a random element of $\mathbb{G}_2$ from the adversaries view and the message contains no information about M_ν.

Phase 2. The simulator acts exactly as it did in Phase 1.

Guess. $\mathcal{A}$ will submit a guess ν' of ν. If $\nu = \nu'$ the simulator will output $\mu' = 0$ to indicate that it was given a BDH-tuple otherwise it will output $\mu' = 1$ to indicate it was given a random 4-tuple.

As shown in the construction the simulator's generation of public parameters and private keys is identical to that of the actual scheme.

In the case where $\mu = 1$ the adversary gains no information about ν. Therefore, we have $\Pr[\nu \neq \nu' | \mu = 1] = \frac{1}{2}$. Since the simulator guesses $\mu' = 1$ when $\nu \neq \nu'$, we have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$.

If $\mu = 0$ then the adversary sees an encryption of M_ν. The adversary's advantage in this situation is ϵ by definition. Therefore, we have $\Pr[\nu = \nu' | \mu = 0] = \frac{1}{2} + \epsilon$. Since the simulator guesses $\mu' = 0$ when $\nu = \nu'$, we have $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon$.

The overall advantage of the simulator in the Decisional BDH game is $\frac{1}{2}\Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2}(\frac{1}{2} + \epsilon) + \frac{1}{2}\frac{1}{2} - \frac{1}{2} = \frac{1}{2}\epsilon$. □

7 Conclusions

We introduced the concept of Fuzzy Identity Based Encryption, which allows for error-tolerance between the identity of a private key and the public key used to encrypt a ciphertext. We described two practical applications of Fuzzy-IBE of encryption using biometrics and attribute-based encryption.

We presented our construction of a Fuzzy IBE scheme that uses set overlap as the distance metric between identities. Finally, we proved our scheme under the Selective-ID model by reducing it to an assumption that can be viewed as a modified version of the Bilinear Decisional Diffie-Hellman assumption.

This work motivates a few interesting open problems. The first is whether it is possible to create a Fuzzy IBE scheme where the attributes come from multiple authorities. While, it is natural for one authority to certify all attributes that compromise a biometric, in attribute-based encryption systems there will often not be one party that can act as an authority for all attributes. Also, a Fuzzy-IBE scheme that hides the public key that was used to encrypt the ciphertext [1] is intriguing. Our scheme uses set-overlap as a similarity measure between identities. (We note a Hamming-distance construction can also be built using our techniques.) An open problem is to build other Fuzzy-IBE schemes that use different distance metrics between identities.

Acknowledgments

We would like to thank Don Coppersmith, Ed Felten and Philippe Golle, Ari Juels, and the reviewers of Eurocrypt 2005 for providing helpful comments and suggestions.

References

1. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and D. Pointcheval. Key-privacy in public-key encryption. *Lecture Notes in Computer Science*, 2248, 2001.
2. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity based encryption without random oracles. In *Proceedings of the International Conference on Advances in Cryptology (EUROCRYPT '04)*, Lecture Notes in Computer Science. Springer Verlag, 2004.
3. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229. Springer-Verlag, 2001.
4. Xavier Boyen. Reusable cryptographic fuzzy extractors. In *ACM Conference on Computer and Communications Security—CCS 2004*, 2004.
5. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Proceedings of Eurocrypt 2003*. Springer-Verlag, 2003.
6. G.I. Davida, Y. Frankel, and B.J. Matt. On enabling secure applications through off-line biometric identification. *In IEEE Symposium on Privacy and Security*, 1998.
7. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate string keys from biometrics and other noisy data. In *Proceedings of the International Conference on Advances in Cryptology (EUROCRYPT '04)*, Lecture Notes in Computer Science. Springer Verlag, 2004.
8. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 537–554. Springer-Verlag, 1999.
9. Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 28–36. ACM Press, 1999.
10. Fabian Monrose, Michael K. Reiter, Q. (Peter) Li, Daniel Lopresti, and Chilin Shih. Towards voice generated cryptographic keys on resource constrained devices. In *Proceedings of the 11th USENIX Security Symposium*, 2002.
11. Fabian Monrose, Michael K. Reiter, Q. (Peter) Li, and Susanne Wetzel. Cryptographic key generation from voice. In *Proceedings of the IEEE Conference on Security and Privacy*, 2001.
12. Fabian Monrose, Michael K. Reiter, and Susanne Wetzel. Password hardening based on keystroke dynamics. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 73–82. ACM Press, 1999.
13. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *In Proceedings of 40 IEEE Symp. on Foundations of Computer Science*, 1999.
14. Adi Shamir. How to share a secret. *Communications. ACM*, 22(11):612–613, 1979.
15. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53. Springer-Verlag New York, Inc., 1985.
16. Brent Waters. Efficient identity based encryption without random oracles. In *To Appear in Proceedings Eurocrypt 2005*, 2005.
17. Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *ACM Conference on Computer and Communications Security—CCS 2004*, 2004.

Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work

John Kelsey[1] and Bruce Schneier[2]

[1] National Institute of Standards and Technology
john.kelsey@nist.gov
[2] Counterpane Internet Security, Inc.
schneier@counterpane.com

Abstract. We expand a previous result of Dean [Dea99] to provide a second preimage attack on all n-bit iterated hash functions with Damgård-Merkle strengthening and n-bit intermediate states, allowing a second preimage to be found for a 2^k-message-block message with about $k \times 2^{n/2+1} + 2^{n-k+1}$ work. Using RIPEMD-160 as an example, our attack can find a second preimage for a 2^{60} byte message in about 2^{106} work, rather than the previously expected 2^{160} work. We also provide slightly cheaper ways to find multicollisions than the method of Joux [Jou04]. Both of these results are based on *expandable messages*–patterns for producing messages of varying length, which all collide on the intermediate hash result immediately after processing the message. We provide an algorithm for finding expandable messages for any n-bit hash function built using the Damgård-Merkle construction, which requires only a small multiple of the work done to find a single collision in the hash function.

1 Introduction

The security goal for an n-bit hash function is that collisions require about $2^{n/2}$ work, while preimages and second preimages require about 2^n work. In [Dea99], Dean demonstrated that this goal could not be accomplished by hash functions whose compression functions allowed the easy finding of fixed points, such as MD5 [Riv92] and SHA1 [SHA02]. In this paper, we use the multicollision-finding result of [Jou04] to demonstrate that the standard way of constructing iterated hash functions (the Damgård-Merkle construction) cannot meet this goal, regardless of the compression function used. Thus, hash functions such as RIPEMD-160 [DBP96] and Whirlpool [BR00] (when used with a full 512-bit result) provide less than the previously-expected amount of resistance to second-preimage attacks, just as do hash functions like SHA1.

For a message of 2^k message blocks, we provide a second preimage attack requiring about $k \times 2^{n/2+1} + 2^{n-k+1}$ work. Like Dean's attack, ours is made possible by the notion of *expandable messages*–patterns of messages of different lengths which all yield the same intermediate hash value after processing them. These expandable messages do not directly yield collisions on the whole hash function

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 474–490, 2005.

because of the length padding done at the end of modern hash functions, and in any event are no easier to find than collisions. However, they allow second preimages and multicollisions to be found much more cheaply than had previously been expected. This result may be compared with an earlier generic preimage attack by Merkle: the attacker is given 2^k distinct n-bit hash outputs, and expects to find a preimage for one of the outputs with about 2^{n-k} work, but has no ability to choose which of the outputs is to be matched. (Note that Merkle's attack is truly generic, in that it applies to any hash fuction with an n-bit output, even a random oracle.)

Our attack, like the earlier attack of Dean, probably has no practical impact on the security of any system currently relying upon a hash function such as SHA1, Whirlpool, or RIPEMD-160. This is true because the attack is always at least as expensive as collision search on the hash function, and because the difficulty of the attack grows quickly as the message gets shorter. For example, a 160-bit hash function like SHA1 or RIPEMD-160 requires about 2^{128} work to find a second preimage for a 2^{38}-byte message, and a target message of only one megabyte (2^{20} bytes) requires about 2^{146} work to find a second preimage. Also, the attack only recovers second preimages–it doesn't allow an attacker to invert the hash function.

The significance of our result is in demonstrating another important way in which the behavior of the hash functions we know how to construct differs from both the commonly claimed security bounds of these functions, and from the random oracles with which we often model them. When combined with the recent results of Joux [Jou04], our results raise questions about the usefulness of the widely-used Damgård-Merkle construction for hash functions where attackers can do more than $2^{n/2}$ work.

The remainder of the paper is organized as follows: First, we discuss basic hash function constructions and security requirements. Next, we demonstrate a generic way to find expandable messages, and review the method of Dean. We then demonstrate how these expandable messages can be used to violate the second preimage resistance of nearly all currently specified cryptographic hash functions with less than 2^n work. Finally, we demonstrate an even more efficient (albeit much less elegant) way to find multicollisions than the method of Joux, using Dean's fixed-point-based expandable messages. We end with a discussion of how this affects our understanding of iterated hash function security.

2 Hash Function Basics

In 1989, Merkle and Damgård [Mer89, Dam89] independently provided security proofs for the basic construction used for almost all modern cryptographic hash functions[1]. Here, we describe this construction[2], and its normal security claims.

[1] These functions date back to Rabin, and were widely used by hash function designers throughout the 1980s [Pre05, MvOV96].

[2] Damgård proposed two methods for constructing hash functions. This paper addresses only the more commonly used one, which was independently invented by Merkle.

A hash function with an n-bit output is expected to have three minimal security properties. (In practice, a number of other properties are expected, as well.)

1. Collision-resistance: An attacker should not be able to find a pair of messages $M \neq M'$ such that $\text{hash}(M) = \text{hash}(M')$ with less than about $2^{n/2}$ work.
2. Preimage-resistance: An attacker given an output value Y in the range of hash should not be able to find an input X from its domain so that $Y = \text{hash}(X)$ with less than about 2^n work.
3. Second preimage-resistance: An attacker given one message M should not be able to find a second message, M' to satisfy $\text{hash}(M) = \text{hash}(M')$ with less than about 2^n work.

A collision attack on an n-bit hash function with less than $2^{n/2}$ work, or a preimage or second preimage attack with less than 2^n work, is formally a break of the hash function. Whether the break poses a practical threat to systems using the hash function depends on specifics of the attack.

Following the Damgård-Merkle construction, an iterated hash function is built from a fixed-length component called a compression function, which takes an n-bit input chaining value and an m-bit message block, and derives a new n-bit output chaining value. In this paper, $F(H, M)$ is used to represent the application of this compression function on hash chaining variable H and message block M.

In order to hash a full message, the following steps are carried out:

1. The input string is padded to ensure that it is an integer multiple of m bits in length, and that the length of the original, unpadded message appears in the last block of the padded message.
2. The hash chaining value $h[i]$ is started at some fixed IV, $h[-1]$, for the hash function, and updated for each successive message block $M[i]$ as
$$h[i] = F(h[i-1], M[i])$$
3. The value of $h[i]$ after processing the last block of the padded message is the hash output value.

This construction gives a reduction proof: If an attacker can find a collision in the whole hash, then he can likewise find one in the compression function. The inclusion of the length at the end of the message is important for this reduction proof, and is also important for preventing a number of attacks, including long-message attacks [MvOV96].

Besides the claimed security bounds, there are two concepts from this brief discussion that are important for the rest of this paper:

1. A message made up of many blocks, $M[0, 1, 2, ..., 2^k - 1]$, has a corresponding sequence of intermediate hash values, $h[0, 1, 2, ..., 2^k - 1]$.
2. The padding of the final block includes the length, and thus prevents collisions between messages of different lengths in the intermediate hash states from yielding collisions in the full hash function.

3 Finding Expandable Messages

An expandable message is a kind of multicollision, in which the colliding messages have different lengths, and the message hashes collide in the *input* to the last compression function computation, before the length of the message is processed. Consider a starting hash value $h[-1]$. Then an "expandable message" from $h[-1]$ is a pattern for generating messages of different lengths, all of which yield the same intermediate hash value when they are processed by the hash, starting from $h[-1]$, without the final padding block with the message length being included. In the remainder of the paper, an expandable message that can take on any length between a and b message blocks, inclusive, will be called an (a, b)-expandable message.

3.1 Dean's Fixed-Point Expandable Messages

In [Dea99], there appears a technique for building expandable messages when fixed points can easily be found in the compression function[3]. 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 $h[i-1] = F(h[i-1], M[i])$. Compression functions based on the Davies-Meyer construction [MvOV96], such as the SHA family [SHA02], MD4, MD5 [Riv92], and Tiger [AB96], have easily found fixed points. Similarly, Snefru [Mer90] has easily found fixed points. Techniques for finding these fixed points for compression functions based on the Davies-Meyer construction appear in [MOI91], and are briefly discussed in an appendix to this paper, along with techniques for finding fixed points in Snefru. Note that these techniques produce a pair $(h[i-1], M[i])$, but allow no control over the value of $h[i-1]$.

We can construct an expandable message using fixed points for about twice as much work as is required to find a collision in the hash function. This is done by first finding about $2^{n/2}$ randomly-selected fixed points for the compression function, and then trying first message blocks until one leads from the initial hash value to one of the fixed points.

ALGORITHM: MakeFixedPointExpandableMessage($h[in]$)

Make an expandable message from initial hash value $h[in]$, using a fixed point finding algorithm.

Variables:

1. $h[in]$ = initial chaining value for the expandable messages.
2. FindRandomFixedPoint() = an algorithm returning a pair $(h[i], M[i])$ such that $h[i] = F(h[i], M[i])$.
3. A, C = two lists of hash values.
4. B, D = two lists of message blocks.
5. i, j = integers.

[3] We were made aware of Dean's work by a comment from one of the anonymous referees.

6. $M(i)$ = a function that produces a unique message block for each integer i less than 2^n.
7. n = width of hash function chaining value and output.

Steps:

1. Construct a list of $2^{n/2}$ fixed points:
   - For $i = 0$ to $2^{n/2} - 1$:
     - h, m = FindRandomFixedPoint()
     - $A[i] = h$
     - $B[i] = m$
2. Construct a list of $2^{n/2}$ hash values we can reach from $h[-1]$:
   - For $i = 0$ to $2^{n/2} - 1$:
     - $h = F(h[in], M(i))$
     - $C[i] = h$
     - $D[i] = M(i)$
3. Find a match between lists A and C; let i, j satisfy $A[i] = C[j]$.
4. Return expandable message $(D[j], B[i])$.

Work: About $2^{n/2+1}$ compression function computations, assuming $2^{n/2+1}$ memory.

If an n-bit hash function has a maximum of 2^k blocks in its messages, then this technique takes about $2^{n/2+1}$ work to discover $(1, 2^k)$-expandable messages. Producing a message of the desired length is trivial, consisting of one copy of the starting message block, and as many copies of the fixed-point message block as necessary to get a full message of the desired length.

ALGORITHM: ProduceMessageFP(R, X, Y)

Produce a message of desired length from the fixed-point expandable messages.

Variables:

1. R = the desired length in message blocks; must be at least one and no more than the maximum number of message blocks supported by the hash.
2. X = the first message block in the expandable message.
3. Y = the second (repeatable) block in the expandable message.

Steps:

1. $M = X$.
2. For $i = 0$ to $R - 2$:
   - $M = M||Y$
3. Return M.

Work: Negligible work, about R steps.

3.2 A Generic Technique: Multicollisions of Different Lengths

Finding an expandable message for any compression function with n-bit intermediate hash values takes only a little more work than finding a collision in the hash function. This technique is closely related to the technique for finding k-collisions in iterated hash functions from Joux.

In Joux's technique, a sequence of single-message-block collisions is found, and then pasted together to provide a large number of different messages of equal length that lead to the same hash value. In our technique, a sequence of collisions between messages of *different* lengths is found, and pasted together to provide a set of messages that can take on a wide range of different lengths without changing the resulting intermediate hash value—an expandable message.

Finding a Collision on Two Messages of Different Lengths. Finding an expandable message requires the ability to find many pairs of messages of different specified lengths that have the same resulting intermediate hash value. Finding such a pair is not fundamentally different than finding a pair of equal-length messages that collide: The attacker who wants a collision between a one-block message and an α-block message constructs about $2^{n/2}$ messages of length 1, and about the same number of length α, and looks for a collision. For efficiency, the attacker chooses a set of α-block messages whose hashes can be computed about as efficiently as the same number of single-block messages.

ALGORITHM: FindCollision(α, h_{in})

Find a collision pair with lengths 1 and blocks, starting from h_{in}.

Variables:

1. α = desired length of second message.
2. A, B = lists of intermediate hash values.
3. q = a fixed "dummy" message used for getting the desired length.
4. h_{in} = the input hash value for the collision.
5. h_{tmp} = intermediate hash value used in the attack.
6. $M(i)$ = the ith distinct message block used in the attack.
7. n = width of hash function chaining value and output in bits.

Steps:

1. Compute the starting hash for the α-block message by processing $\alpha - 1$ dummy message blocks:
   - $h_{tmp} = h_{in}$.
   - For $i = 0$ to $\alpha - 2$:
     - $h_{tmp} = F(h_{tmp}, q)$
2. Build lists A and B as follows:
   - for $i = 0$ to $2^{n/2} - 1$:
     - $A[i] = F(h_{in}, M(i))$
     - $B[i] = F(h_{tmp}, M(i))$
3. Find i, j such that $A[i] = B[j]$
4. Return colliding messages $(M(i), q||q||...||q||M(j))$, and the resulting intermediate hash $F(h_{in}, M(i))$.

Work: $\alpha - 1 + 2^{n/2+1}$ compression function calls

Building a Full $(k, k+2^k-1)$-expandable message. We can use the above algorithm to construct expandable messages that cover a huge range of possible lengths, in a technique derived from the multicollision-finding technique of [Jou04]. We first find a colliding pair of messages, where one is of one block, and the other of $2^{k-1}+1$ blocks. Next, we find a collision pair of length either 1 or $2^{k-2}+1$, then 1 or $2^{k-3}+1$, and so on, until we reach a collision pair of length 1 or length 2. The result is a list of pairs of message components of different lengths, which lead to the same intermediate hash after processing them. The first such pair allows a choice of adding 2^{k-1} blocks to the expanded message, the second allows a choice of adding 2^{k-2} blocks, and so on. Thus, expanding the message is just writing the difference between the desired length and the number of message components in binary, and using each bit in that binary string to choose the corresponding short or long message component to include.

ALGORITHM: MakeExpandableMessage(h_{in}, k)
Make a $(k, k+2^k-1)$-expandable message.

Variables:

1. h_{tmp} = the current intermediate hash value.
2. C = a list of pairs of messages of different lengths; $C[i][0]$ is the first message of pair i, while $C[i][1]$ is that pair's second message.

Steps:

1. Let $h_{tmp} = h_{in}$.
2. For $i = 0$ to $k-1$:
   - $(m_0, m_1, h_{tmp}) = \text{FindCollision}(2^i+1, h_{tmp})$
   - $C[k-i-1][0] = m_0$
   - $C[k-i-1][1] = m_1$
3. Return the list of message pairs C.

Work: $k \times 2^{n/2+1} + 2^k \approx k \times 2^{n/2+1}$ compression function calls.

At the end of this process, we have an $k \times 2$ array of messages, for which we have done approximately $2^k + k \times 2^{n/2+1}$ compression function computations, and with which we can build a message consisting of between k and $k+2^k-1$ blocks, inclusive, without changing the result of hashing the message until the final padding block.

Producing a Message of Desired Length. Finally, there is a simple algorithm for producing a message of desired length from an expanded message. This amounts to simply including the different-length pieces based on the bit pattern of the desired length.

ALGORITHM: ProduceMessage(C, k, L)
Produce a message of length L, if possible, from the expandable message specified by (C, k).

Variables:

1. L = desired message length.
2. k = parameter specifying that C contains a $(k, k + 2^k - 1)$-expandable message.
3. C = a $k \times 2$ array of message fragments of different lengths.
4. M = the message to be constructed.
5. T = a temporary variable holding the remaining length to be added.
6. $S[0..k-1]$ = a sequence of bits from T.
7. i = an integer counter.

Steps:

1. Start with an empty message $M = \emptyset$.
2. If $L > 2^k + k - 1$ or $L < k$, return an error condition.
3. Let $T = L - k$.
4. Let S = the bit sequence of T, from low-order to high-order bits.
5. Concatenate message fragments from the expandable message together until we get the desired message length. Note that this is very similar to writing T in binary.
   - for $i = 0$ to $k - 1$:
     - if $S[i] = 0$ then $M = M||C[i][0]$
     - else $M = M||C[i][1]$
6. Return M.

Work: Negligible (about k table lookups and string copying operations).

The result of this is a message of the desired length, with the same hash result before the final padding block is processed as all the other messages that can be produced from this expandable message.

3.3 Variants

The expandable messages found by both of these methods can start at any given hash chaining value. As a result, we can build expandable messages with many useful properties:

1. The expandable message can start with any desired prefix.
2. The expandable message can end with any desired suffix.
3. While both algorithms given here for finding expandable messages assume complete freedom over choice of message block, a variant of the generic method can be used even if the attacker is restricted to only two possible values for each message block.
4. The fixed-point method requires about $2^{n/2}$ possible values for each message block, but this is sufficiently flexible that for existing hash functions, it can typically be used with only ASCII text, legitimate sequences of Pentium opcodes, etc.
5. The multicollision technique from Joux allows an attacker to discover 2^k messages with the same hash for an n-bit iterated hash function, using only about $k \times 2^{n/2}$ compression functions of work. This technique can be used to make a set of 2^k expandable messages which all yield the same hash output. The full power of combining these techniques remains to be investigated.

4 Using Expandable Messages to Find Second Preimages

An n-bit hash function is supposed to resist second preimage attacks up to about 2^n work. That is, given one message M, the attacker ought to have to spend about 2^n work to find another message that has the same hash value as output.

4.1 The Long Message Attack

Here is a general (and previously known) way to violate the second-preimage resistance of a hash function without Damgård-Merkle strengthening [MvOV96]: Start with an extremely long message of $2^R + 1$ blocks. An attacker who wishes to find another message that hashes to the same value with a 160-bit hash function can do so by finding a message block M_{link} such that, from the IV of the hash, $h[-1]$, $h^* = F(h[-1], M_{link})$ yields a value h^* that matches one of the intermediate values of the hash function in processing the long message. Since the message has 2^R such intermediate values, the attacker expects to need to try only about 2^{160-R} message blocks to get a match. That is, when $R = 64$, the attacker has 2^{64} available target values, so each message block he tries has about a 2^{-96} chance of yielding the same hash output as some intermediate hash value from the target message. The result is a shorter message, which has the same hash output *up until the final block is processed.*

The length padding at the end of the Damgård-Merkle construction foils this attack. Note that in the above situation, the attacker has a message that is shorter than the 2^{55}-block target message, which leads to the same intermediate hash value . But now, the last block has a different length field, and so the attack fails—the attacker can find something that's *almost* a second preimage, but the length block changes, and so the final hash output is different.

4.2 Long-Message Attacks with Expandable Messages

Using expandable messages, we can bypass this defense, and carry out a second-preimage attack despite the length block at the end. This attack was first discovered by Dean [Dea99]. We start with a long message as our target for a second preimage, find an expandable message which will provide messages over a wide range of lengths, and then carry out the long-message attack from the end of that expandable message. We then expand the expandable message to make up for all the message blocks that were skipped by the long message attack, yielding a new message of the same length as the target message, with the same hash value.

ALGORITHM: LongMessageAttack(M_{target})
Find the second preimage for a message of $2^k + k + 1$ *blocks.*

Variables:

1. M_{target} = the message for which a second preimage is to be found.
2. M_{link} = a message block used to link the expandable message to some point in the target message's sequence of intermediate hash values.

3. A = a list of intermediate hash values
4. h_{exp} = intermediate chaining value from processing an expandable message.

Steps:

1. C = MakeExpandableMessage(k)
2. h_{exp} = the intermediate hash value after processing the expandable message in C.
3. Compute the intermediate hash values for M_{target}:
   - $h[-1]$ = the IV for the hash function
   - $m[i]$ = the ith message block of M_{target}.
   - $h[i] = F(h[i-1], m[i])$, the ith intermediate hash output block. Note that h will be organized in some searchable structure for the attack, such as a hash table, and that elements $h[0, 1, ..., k]$ are excluded from the hash table, since the expandable message cannot be made short enough to accommodate them in the attack.
4. Find a message block that links the expandable message to one of the intermediate hash values for the target message after the kth block.
   - Try linking messages M_{link} until $F(h_{exp}, M_{link}) = h[j]$ for some $k + 1 \leq j \leq 2^k + k + 1$.
5. Use the expandable message to produce a message M^* that is $j-1$ blocks long.
6. Return second preimage $M^*||M_{link}||m[j + 1]||m[j + 2]...m[2^k + k + 1]$ (if $j = 2^k + k + 1$, then no original message blocks are included in the second preimage).

Work: The total work done is the work to find the expandable message plus the work to find the linking message.

1. For the generic expandable message-finding algorithm, this is $k \times 2^{n/2+1} + 2^{n-k+1}$ compression function calls.
2. For the fixed-point expandable message-finding algorithm, this is $3 \times 2^{n/2+1} + 2^{n-k+1}$

The longer the target message, the more efficient the attack relative to a brute-force preimage search, until the search for the expandable message becomes more expensive than the long-message attack. For SHA1 and SHA256, the maximum allowed message length is $2^{64} - 1$ bits, which translates to about 2^{55} 512-bit blocks of message. For SHA384 and SHA512, the maximum allowed message length is $2^{128} - 1$ bits, which translates to about 2^{118} 1024-bit blocks of message. Let 2^R be the maximum number of message blocks allowed by the hash function. The total work of the generic expandable-message form of the attack is then $R \times 2^{n/2+1} + 2^{n-R+1}$ compression function calls.

An Illustration. To illustrate this, consider a second preimage attack on the RIPEMD-160 hash function [DBP96]. The longest possible message for RIPEMD-160 is $2^{64} - 1$ bits, which translates into just under 2^{55} blocks. For simplicity, we will assume the target message is $2^{54} + 54 + 1$ message blocks (about 2^{60} bytes) long.

1. Receive the target message and compute and store all the intermediate hash values.
2. Produce a $(1, 54 + 2^{54})$-expandable message. This requires about 54×2^{81} compression function computations.
3. Starting from the end of the expandable message, we try about 2^{106} different message blocks, until we find one whose hash output is the same as one of the last $54 + 2^{54}$ intermediate hash values of the target message. This requires computing about 2^{106} compression functions on average.
4. Expand the expandable message to compensate for the message blocks of the target message skipped over, and thus produce a second preimage. This takes very little time.

Summary of the Attack. The long-message attack can be summarized as follows: For a target message substantially less than $2^{n/2}$ blocks in length, the work is dominated by the long message attack. Thus, a second preimage attack on a 2^k-block message takes about 2^{n-k+1} compression function computations, assuming abundant memory.

4.3 Variations on the Attack

Some straightforward variations of this attack are also possible, drawing from the variations available to the expandable messages. For example, the algorithms for producing an expandable message work from any starting hash value, and are not affected by the message blocks that come after the expanded message. Thus, this attack can be used to "splice together" two very long messages, with an expandable part in the middle. Similarly, if it is important that the second preimage message start with the same first few hundred or thousand message blocks as the target message, or end with the same last few hundred or thousand blocks, this can easily be accommodated in the attack. Another variation is available by using Joux's multicollision-finding trick, or the related ones described below: By setting up the expandable message to be a 2^u-multicollision, we can find 2^u distinct second preimages for a given long message, without adding substantial cost to the attack. Additionally, keyed constructions that leave the attacker with offline collision search abilities are vulnerable to the attack; for example, the "suffix mac" construction [MvOV96], $\text{MAC}_K(X) = \text{Hash}(X||K)$ is vulnerable to a second preimage attack, as well as the much more practical, previously-known collision attack.

Low-Memory and Parallel Versions of the Attack. These methods for finding expandable messages assume unlimited memory. In the real world, memory is limited, and bandwidth between processing units and memory units is likewise limited. This doesn't raise a difficulty to the attack. For n-bit hash functions whose maximum input size in message blocks is substantially less than $2^{n/2}$, the parallel collision search techniques of [vOW96, vOW99] allow both our generic attack and the fixed-point attack of Dean to go forward at approximately the stated cost; the search for a linking message (the long message attack) dominates the work.

4.4 The Attack in Perspective

Our attack allows the finding of a second preimage on a 2^k block long target message with a certainty of success. A previously known attack originally noted by Merkle is somewhat similar in spirit [MvOV96, Pre05]: an attacker is given 2^k candidate target messages, and finds one preimage with 2^{n-k} work. While that attack wasn't able to find a second preimage for a specific desired message, it makes our result and the earlier result of Dean somewhat less surprising. It is worth noting that Merkle's result applies to *any* n-bit hash function, even one constructed from random oracles.

Our attack differs from that of Dean only in its universality–Dean's attack applies only to hash functions whose compression functions allow easy finding of second preimages, whereas ours apply to any iterated hash function with an n-bit intermediate hash value.

5 Expandable Messages and Multicollisions

In [Jou04], Joux demonstrates a beautiful way to produce a large number of messages that collide for an iterated hash function, with only a little more work than is needed to find a single pair of messages that collide. Here, we demonstrate ways to use expandable messages to find multicollisions, and ways to combine the Joux technique with expandable messages to add flexibility to the structure of the multicollisions.

We construct a multicollision by concatenating two or more expandable messages, and then varying the length of each so that the sum of their lengths stays the same. For example, if we concatenate a (1,1024)-expandable message with another (1,1024)-expandable message, we get a 1024-collision of 1025 block long messages. By concatenating a large number of such messages, we can get a much larger multicollision.

5.1 Multicollisions Using Fixed Points

Using fixed-point expandable messages, multicollisions which are much cheaper than those found by Joux are available. Recall that for an n-bit hash function, finding a fixed-point expandable message which is expandable up to the maximum message length of the hash function costs about $2^{n/2+1}$ compression function computations.

Consider a 160-bit hash function with a maximum of 2^{55} message blocks. Now, a very simple 2^{55}-collision is available for about $2^{82} = 4 \times 2^{80}$ work, as opposed to 55×2^{80} work–this is constructed by concatenating two fixed-point expandable messages, and always making the sum of their lengths 2^{55} blocks. Concatenating three such expandable messages produces a 2^{107}-collision, and so on, following the rule that a multicollision consisting of K expandable messages in a hash function with a maximum length of R blocks produces $\binom{R}{K-1}$-multicollisions with about $K \times 2^{81}$ work.

These multicollisions are of unreasonable length, but they're generally cheaper than Joux' multicollisions. At more reasonable lengths, they're still interesting, but they become more expensive than Joux' multicollisions. For example, concatenating ten expandable messages together and limiting message length to 1034 blocks total, we get about a 2^{80}-multicollision using this technique; for the same cost, Joux' technique would give a 2^{1024}-multicollision.

5.2 Using Generic Expandable Messages

The cost of finding a single fixed-point expandable message is within a factor of two of the cost of finding a single collision in Joux' scheme. The cost of finding a generic $(1, K)$-expandable message is about $\lg(K) \times 2^{n/2}$. This means that in general, generic expandable messages cannot be used to make multicollisions cheaper than those of Joux.

5.3 Combining with Joux

Finally, it is possible to combine Joux multicollisions with expandable-message multicollisions. This allows multicollisions to be constructed that look quite different from the Joux multicollisions, and are somewhat more flexible in structure. This may allow Joux attacks to go forward even on cascaded constructions that attempt to foil his attack.

As an example, a multicollision may be formed by alternating $(1, 2)$-expandable messages and individual collisions as sought by Joux' method, with a final $(10, 1024)$-expandable message at the end. This permits the individual colliding message blocks to appear at different positions in different messages, without altering the final hash value.

6 Conclusions and Open Questions

In this paper, we have described a generic way to carry out long-message second preimage attacks, despite the Damgård-Merkle strengthening done on all modern hash functions.

These attacks are theoretical because 1) they require more work than is necessary to find collisions on the underlying hash functions, and 2) the messages for which second preimages may be found are generally impractically long. However, they demonstrate some new lessons about hash function design:

1. An n-bit iterated hash function provides fundamentally different security properties than a random oracle with an n-bit output. This was demonstrated in one way by Joux in [Jou04], and by another here.
2. An n-bit iterated hash function begins to show some surprising properties as soon as an attacker can do the work necessary to find collisions in the underlying compression function.
3. An n-bit iterated hash function cannot support second-preimage resistance at the n-bit security level, as previously expected, for long messages.

4. Easily found fixed points in compression functions (such as those based on the Davies-Meyer construction) allow an even more powerful second-preimage attack described in [Dea99].

The important lesson here is that the standard construction of iterated hashes from Merkle and Damgård does not provide all the protection we might expect against attackers that can do more than $2^{n/2}$ compression function computations. In some sense, the hash function is "brittle," and begins to lose its claimed security properties very quickly once the attacker can violate its collision resistance by brute force.

We believe these results, when combined with those of Dean and Joux, require a rethinking of what security properties are expected of an iterative hash function with an n-bit intermediate state. We see three sensible directions for this rethinking to take:

1. A widespread consensus that an n-bit iterated hash function should never be expected to resist attacks requiring more than $2^{n/2}$ operations. This would invalidate current uses of hash functions in cryptographic random-number generation, as in [KSF99, DHL02, Bal98], key derivation functions as described in [AKMZ04, NIST03, X963], and many other applications, and seems the least palatable outcome.
2. A clear theoretical treatment of the limits that exist for n-bit hash functions, and precisely what attacks more demanding than collision search they may be expected to resist. (For example, none of these recent results appear to be applicable when the attacker cannot do offline collision search. Similarly, these attacks do not apply when only a single message block is being processed. Perhaps these observations can be formalized.)
3. New constructions for hash-function round functions. For example, XORing in a monotomic counter as part of the round function would resist the attacks in this paper.
4. New constructions for hash functions in the vein of [Luc04], which maintain much more than n bits of intermediate state in order to make collision attacks on intermediate states harder (require 2^n work).

We believe that the region between $2^{n/2}$ and 2^n is a rich area for the cryptanalysis of iterated hash functions, and expect to see other research results in the future. Absent a solid theoretical treatment of the security properties of n-bit iterative hashes along the lines of [PGV93] and [BRS02], expanded to deal thoroughly with the full hash construction, at this point it is difficult to justify using them in applications requiring more than $n/2$ bits of security for messages longer than one block with any confidence.

We hope this work spurs such a treatment, as well as further cryptanalysis.

Acknowledgements

The authors wish to thank Bill Burr, Morris Dworkin, Matt Fanto, Niels Ferguson, Phil Hawkes, Julie Kelsey, Ulrich Kühn, Arjen Lenstra, Stefan Lucks, Bart

Preneel, Vincent Rijmen, David Wagner, and the anonymous referees for many useful comments and discussions on the results in this paper.

References

[AB96] Anderson and Biham, "Tiger—A Fast New Hash Function," in proceedings of FSE96, Springer-Verlag, 1996.

[AKMZ04] Adams, Kramer, Mister, and Zuccherato, "On the Security of Key Derivation Functions," *Proceedings of the 7th Information Security Conference (ISC '04)*, Palo Alto, CA, USA, Springer-Verlag, 2004 (to appear).

[Bal98] Baldwin, "Preliminary Analysis of the BSAFE 3.x Pseudorandom Number Generators," *RSA Laboratories Bulletin No. 8*, RSA Laboratories, 1998.

[BR00] P. S. L. M. Barreto and V. Rijmen, "The Whirlpool Hashing Function", *First open NESSIE Workshop*, Leuven, Belgium, 13–14 November 2000.

[BRS02] Black, Rogaway, and Shrimpton, "Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV," *Advances in Cryptology–Crypto 02 Proceedings*, Springer-Verlag, 2002.

[BS93] Biham and Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

[DBP96] Dobbertin, Bosselaers, and Preneel, "RIPEMD-160, A Strengthened Version of RIPEMD," in *Fast Software Encryption 1996*, Springer-Verlag, 1996

[Dea99] Richard Drews Dean, *Formal Aspects of Mobile Code Security*, Ph.D Dissertation, Princeton University, January 1999.

[Dam89] Damgård, "A Design Principle for Hash Functions," *Advances in Cryptology–Crypto 89 Proceedings*, Springer-Verlag, 1989.

[DHL02] Desai, Hevia, and Yin, "A Practice-Oriented Treatment of Pseudorandom Number Generators," *Advances in Cryptology–Eurocrypt 02 Proceedings*, Springer-Verlag, 2002.

[Jou04] Joux, "Multicollisions in Iterated Hash Functions. Applications to Cascaded Constructions," *Advances in Cryptology–Crypto 2004 Proceedings*, Springer-Verlag, 2004.

[KSF99] Kelsey, Schneier, and Ferguson, "Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator," SAC 1999.

[Luc04] Lucks, "Design Principles for Iterated Hash Functions," IACR preprint archive, `http://eprint.iacr.org/2004/253.pdf`, 2004.

[Mer89] Merkle, "One Way Hash Functions and DES," *Advances in Cryptology–Crypto 89 Proceedings*, Springer-Verlag, 1989.

[Mer90] Merkle, "A Fast Software One-Way Hash Function," *Journal of Cryptology*, 3(1):43–58, 1990

[MOI91] Miyaguchi, Ohta, Iwata, "Confirmation that Some Hash Functions are Not Collision Free," *Advances in Cryptology–Crypto 90 Proceedings*, Springer-Verlag, 1990.

[MvOV96] Menezes, van Oorschot, Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[NIST03] *NIST Special Publication 800-56, Recommendations on Key Establishment Schemes*, Draft 2.0, Jan 2003, available from `csrc.nist.gov/CryptoToolkit/kms/keyschemes-jan02.pdf`.

[PGV93] Preneel, Govaerts, and Vandewalle, "Hash Functions Based on Block Ciphers: A Synthetic Approach," *Advances in Cryptology–Crypto 93 Proceedings*, Springer-Verlag, 1993.

[Pre05] Preneel, *Personal Communication*, March 2005.

[Riv92] Rivest, "The MD5 Message-Digest Algorithm," RFC1321, April 1992.

[SHA02] National Institute of Standards and Technology, *Secure Hash Standard,*' FIPS180-2, August 2002.

[vOW99] van Oorschot and Wiener, "Parallel Collision Search with Cryptanalytic Applications," *J. of Cryptology*, 12:1–28, 1999.

[vOW96] van Oorschot and Wiener, "Improving Implementable Meet-in-the-Middle Attacks by Orders of Magnitude," *Advances in Cryptology–Crypto 96 Proceedings*, Springer-Verlag, 1996.

[X963] "ANSI X9.63—Public Key Cryptography for the Financial Services Industry: Key Agreement and Transport Using Elliptic Curve Cryptography," American Bankers Association, 1999. Working Draft.

A Finding Fixed Points Efficiently in Many Compression Functions

Finding fixed points in many hash compression functions is simple.

A.1 Davies-Meyer

Most widely used hash functions have compression functions designed around very large block-cipher-like constructions, following the general Davies-Meyer model. For the SHA and MD4/MD5 families, as well as Tiger, if $E(K, X)$ is a very wide block cipher, with K the key and X the value being encrypted, then the compression function is:

$$F(H, M) = E(M, H) + H$$

for some group operation "+". For these compression functions, it is possible to compute the inverse of this block-cipher-like construction, which we can denote as $E^{-1}(K, X)$. This makes it possible to find fixed points in a simple way, as discussed in [MOI91] and [PGV93]:

1. Select a message M.
2. Compute $H = E^{-1}(M, 0)$.
3. The result gives a fixed point: $F(H, M) = H$.

A property of this method for finding fixed points is that the attacker is able to choose the message, but he has no control whatsoever over the hash value that is a fixed point for a given message. Also note that for these hash functions, each message block has exactly one fixed point.

A.2 Snefru

Snefru is derived from a block-cipher-like operation that operates on a much larger block than the hash output, and which effectively has a fixed "key." Let $E(X)$ be this fixed "encryption" of a block. Further, let n be the hash block size, m be the message block size, $\text{lsb}_n(X)$ be the least significant n bits of X, and $\text{msb}_n(X)$ be the most significant n bits of X. Note that $E(X)$ operates on $n + m$-bit blocks.

The compression function is derived from $E(X)$:

$$F(H, M) = \text{lsb}_n(E(H||M)) + H$$

where the hash input and output are each n bits wide, and where $\text{lsb}_x(Y)$ represents the least significant x bits of the value Y. We can find fixed points for Snefru-like compression functions as follows, letting $E^{-1}(X)$ be the inverse of $E(X)$ once again:

1. Choose any X whose least significant n bits are 0.
2. Compute $Y = E^{-1}(X)$.
3. Let $H = \text{msb}_n(Y)$ and $M = \text{lsb}_m(Y)$.
4. The result gives a fixed point: $F(H, M) = H$.

This method gives the attacker no control over the message block. Unlike the Davies-Meyer construction, there is no guarantee that a given message block has even one fixed point; we would expect for some message blocks to have many, and for others to have none.

Note that the Snefru construction could easily be altered to make fixed points very hard to find, when the size of the message and hash blocks are equal, by the compression function as:

$$F(H, M) = \text{lsb}_n(E(H||M)) + H + M$$

or

$$F(H, M) = \text{lsb}_n(E(H||M)) + H + M + \text{msb}_m(E(H||M))$$

Also note that many other compression function constructions, such as the Miyaguchi-Preneel construction used by Whirlpool and N-Hash and the construction used by RIPEMD and RIPEMD-160, do not appear to permit a generic method for finding fixed points.

Predicting and Distinguishing Attacks on RC4 Keystream Generator

Itsik Mantin

NDS Technologies, Israel
imantin@nds.com

Abstract. In this paper we analyze the statistical distribution of the keystream generator used by the stream ciphers RC4 and RC4A. Our first result is the discovery of statistical biases of the digraphs distribution of RC4/RC4A generated streams, where digraphs tend to repeat with short gaps between them. We show how an attacker can use these biased patterns to distinguish RC4 keystreams of 2^{26} bytes and RC4A keystreams of $2^{26.5}$ bytes from randomness with success rate of more than 2/3. Our second result is the discovery of a family of patterns in RC4 keystreams whose probabilities in RC4 keystreams are several times their probabilities in random streams. These patterns can be used to predict bits and words of RC4 with arbitrary advantage, e.g., after 2^{45} output words a single bit can be predicted with probability of 85%, and after 2^{50} output words a single byte can be predicted with probability of 82%, contradicting the unpredictability property of PRNGs.

Keywords: RC4, Stream ciphers, Cryptanalysis, Distinguishing attacks, Predicting attacks.

1 Introduction

RC4 is the most widely used stream cipher in software applications. Among numerous applications it is used to protect Internet traffic as part of the SSL and is integrated into Microsoft Windows. It was designed by Ron Rivest in 1987 and kept as a trade secret until it leaked out in 1994. RC4 has a secret internal state which is a permutation of all the $N = 2^n$ possible n bits words, associated with two indices in it, when in practical applications $n = 8$, and thus RC4 has a huge state of $log_2(2^8! \times (2^8)^2) \approx 1700$ bits.

In this paper we explore several classes of RC4 states, and analyze their statistical properties and cryptanalytic significance. RC4 was already proven to contain many patterns with unique statistical behavior and many correlations between the output words and the internal state. The main innovation of our work is in moving the focus of RC4 analysis from consecutive sequences of rounds to non-consecutive ones. We show classes of RC4 partial states that cause unique behavior of the output stream, when the unique patterns are in correlations between output words in distant rounds.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 491–506, 2005.

Our first result is based on analysis of RC4 1-states (partial states are defined later), which cause digraphs to repeat with short gaps in RC4 output stream, e.g., $ABAB$, $ABCAB$, $ABCDAB$, etc, when the relative bias (the ratio between the additional probability and the original probability) of these patterns is approximately $1/N$ for a zero gap ($ABAB$) and gradually decreases when the gap length increases. We show how these patterns can be used to mount a distinguisher of RC4 streams from randomness that requires only 2^{26} output words, about a third of the data needed by the best known distinguisher from [4] for the same success rate of 2/3. In addition, we show that these patterns appear also in RC4A with the same biased probability as in RC4, and describe a slightly less efficient distinguishing algorithm for RC4A.

Our second result is based on new analysis of RC4 predictive states, which are partial states (usually small) that suffice for determining keystream output for several rounds. We define a recyclability property for these states and show that recyclable predictive states have relatively high probability to repeat in shifts of N rounds, creating the same predicted output pattern repetitively. We prove that some of the known predictive states are indeed recyclable and use these observations to extend the significance of every recyclable predictive state from a single short biased pattern to a family of patterns that occur with various probabilities and with various relative biases. The probabilities are lower than the one of the original pattern, whereas the relative biases are significantly larger than the one of the original pattern and can grow to arbitrary values, allowing the attacker to predict output bits with arbitrary advantage. In addition, we discuss how the recyclable states can be used to construct state recovery attacks on RC4 internal state.

The rest of the paper is organized in the following way: In section 2 we describe RC4 and previous results about its security. In Section 3 we present the digraphs repetition pattern, analyze its statistical properties in RC4/RC4A generated streams and discuss how the cryptanalyst can exploit these properties. In Section 4 we define recyclable states and discuss their availability and their cryptanalytic usability. We summarize our work in Section 5.

2 RC4 and Its Security

2.1 Description of RC4

RC4 consists of 2 parts (described in Figure 1): A key scheduling algorithm KSA which turns a random key (whose typical size is 40-256 bits) into an initial permutation S of $\{0, \ldots, N-1\}$, and an output generation part PRGA which uses this permutation to generate a pseudo-random output sequence.

The PRGA initializes two indices i and j to 0, and then loops over four simple operations which increment i as a counter, increment j pseudo randomly, exchange the two values of S pointed to by i and j, and output the value of S pointed to by $S[i] + S[j]$[1].

[1] Here and in the rest of the paper all the additions are carried out modulo N.

KSA($K[0 \dots \ell - 1]$)
Initialization:
    For $i = 0 \dots N - 1$
        $S[i] = i$
    $j = 0$
Scrambling:
    For $i = 0 \dots N - 1$
        $j = j + S[i] + K[i \bmod \ell]$
        $Swap(S[i], S[j])$

PRGA(K)
Initialization:
    $i = 0$
    $j = 0$
    $S = KSA(K)$
Generation loop:
    $i = i + 1$
    $j = j + S[i]$
    Swap($S[i], S[j]$)
    Output $z = S[S[i] + S[j]]$

Fig. 1. The Key Scheduling Algorithm and the Pseudo-Random Generation Algorithm

2.2 Previous Attacks on RC4

Cryptanalysis of RC4 is divided into two main parts, analysis of the initialization of RC4 and analysis of the keystream generation. The first part focuses on the KSA, the PRGA initialization and the integration of both, whereas the last focuses on the internal state and the round operation of the PRGA.

The simplicity of the initialization part and the major difference between the amount of hidden information between this part and the keystream generation part attracted a lot of attention in the cryptographic community and indeed numerous significant weaknesses were discovered in this part of many types, including classes of weak keys ([18]), patterns that appear twice and three times the expected probability ([5]), propagation of key patterns through the KSA to the initial permutation and through the PRGA initialization to the prefix of the stream ([6]), modes of operation that allow related key attacks ([17]), partial message recovery ([5]) and full key recovery attacks ([6]) with practical time complexities, statistical biases in different prefixes of the generated stream ([6] and [9]) and analysis of the biased distribution of RC4 initial permutation ([7] and [16]).

The weaknesses that were discovered in [6] where the most distructive ones, as they were translated to practical attack on the usage of RC4 in the security protocols (WEP) of the international standard for wireless LAN communication 802.11b. The discovery of this attack affected the trust of cryptographers and security designers in RC4 and the common practice for using RC4 today includes hardening of the initialization process by truncating some prefix of the keystream (RSA and Ron Rivest recommendation is N words). This hardening neutralizes most of the attacks and weaknesses that were discovered in RC4 initialization.

The analysis of the keystream generation part was far less successful in mounting severe attacks, but still several interesting weaknesses where discovered. Golić ([1]) and Fluhrer and McGrew ([4]) designed distinguishers of RC4 streams from random streams that require $2^{44.7}$ and $2^{30.6}$ keystream words respectively. Several classes of RC4 partial states were defined and analyzed in [4], [5] and [8] as such that create unique patterns in the output stream and allow a viewer of the output stream to recover parts of the internal state with more

than trivial probability (chapter 2 of [16] contains an overview of these classes). The cycles structure of RC4 state progression was also analyzed in [3] and [15], where the last describes short cycles that are unreachable by RC4. [2] and [3] describe state recovery attacks with complexity that is less than the square root of an exhaustive search over all possible states. However, due to the hugeness of the state (1700 bits for $n = 8$), these attacks are completely impractical as they require more than 2^{700} steps.

Two variants of RC4 were recently proposed, both slightly more complex than the original RC4 and are claimed to be more secure than it. RC4A ([9]) was designed by Paul and Preneel and works with two RC4 tables (we describe the algorithm in more details in Section 3.4). The generation stage of RC4A is slightly more efficient than RC4's, but the initialization stage requires at least twice the effort of RC4 initialization. VMPC ([10]) was designed by Zoltak and includes several changes to the initialization (j not initialized to 0), the round operation (different progression for j) and the output generation stage (different calculation of the output index). Maximov described in [13] distinguishers for both variants, requiring 2^{54} data for VMPC and 2^{58} data for RC4A.

The trend of side-channel attacks had not skipped RC4 and efficient fault attacks were described in [11] and [12].

3 The Digraph Repetition Bias

In this section we describe a special scenario that occurs when the value 1 is used to update the index j, analyze the expression of this scenario on the statistical behavior of RC4 output streams and exploit these observations to mount a distinguishing attack on RC4.

We use the notations i_t, j_t and S_t for the indices i and j and the permutation S after round t.

3.1 The Queue Model

The behavior of RC4 permutation was described in [16] as a unique queue with some interesting properties. The queue has N ordered elements (permutation elements) and when one reaches its turn, it is used to update the index j through the function $j \leftarrow j + S[i]$. However, instead of going to the end of the line, the updating element selects a pseudo-random location in the queue (pointed to by j), pushes itself to this location and sends the deprived element that was there to the end of the line. One of the properties of this special queue is that values that were recently used (to update j) are likely to be used again in less than N rounds, whereas values that were not used for N rounds must have been pushed back at least once and thus have lower probability to be used.

The permutation itself changes in two locations within every round and in $2k$ locations (possibly with repetitions) within sequences of k rounds due to the swapping, where half of these changes are in predicted locations and the other half are in pseudo-random locations. Lemma 1 measures the effect of these changes on the permutation entries.

Lemma 1. *Let $\mathcal{I}$ be a set of r permutation locations. Suppose that RC4 is in a state where the predictable course of the index i in the next k rounds does not visit $\mathcal{I}$. Then the probability of the permutation k rounds later to have the same values in $\mathcal{I}$ is approximately $e^{-kr/N}$.*

Proof. The index i does not reach any of the indices in $\mathcal{I}$ and the index j progresses in a pseudo-random manner and will reach each of the the r positions in each of the k rounds with probability $1/N$. Thus failing in these kr trials results with having the set $\mathcal{I}$ untouched. The probability of this event is $(1-1/N)^{kr} \approx e^{-kr/N}$. □

3.2 The Digraph Repetition Scenario

Consider the situation where the value 1 is used to update j, i.e. (in "queue terms"), 1 reaches the head of the line. We will denote the round where that happens as round r and the pair of rounds $[r-1, r]$ as the *origin pair*. Notice that this situation occurs quite often, approximately once in every N rounds. Suppose now that the pseudo-random location selected by 1 after the j update (j_r) is not very far from the head of the line, making it unlikely that this element will be moved from there before being used again to update j. We denote the distant which 1 passes in the swap by the gap $g = j_r - i_r$. And lastly, suppose that when i reaches the new location of 1, j points exactly to the original location of 1. We denote this round and the previous one as the *end pair* $[r+g-1, r+g]$. Our observation is that when that happens, a unique equivalence occurs between the origin pair of rounds and the end pair of rounds, making it likely that they produce the same pair of output words. We formalize this observation in Lemma 2.

Lemma 2. *Suppose that $S_{r-1}[i_r] = 1$ and let $g = j_{r-1} - i_{r-1}$ (the gap). If $j_{r+g-1} = i_{r-1}$ then the probability of the digraph that is outputted in rounds $[r+g-1, r+g]$ to be identical to the digraph outputted in rounds $[r-1, r]$ is bounded from below by $e^{(8-8g)/N}$.*

Proof. We first prove that when everything "goes right" the digraphs are identical and then bound the probability that anything "goes wrong". In Figure 2 we track the internal state during rounds $r, \ldots, r+g$ and show that the same digraphs are outputted in the origin pair and the end pair.

This scenario assumes that the permutation values in locations that affect the swaps and the selection of output words in both pairs of rounds, remain unchanged during the whole process (except for the swaps in the origin and end pairs). The locations i_{r-1}, i_r, i_{r+g-1} and i_{r+g} cannot be reached by i but can be changed by j in any of the $g-2$ intermediating rounds. By using Lemma 1 we get that the probability of 4 elements to survive $g-2$ rounds is $e^{-4(g-2)/N}$. The locations of the output words $S_{r-1}[i_{r-1}] + S_{r-1}[i_{r+g-1}]$ and $S_r[i_r] + S_r[i_{r+g}]$ are arbitrary and may fall in the course of i between the round pairs. Each of these entries need to remain unchanged for g rounds (the first in rounds $r, \ldots, r+g-1$ and the second in rounds $r+1, \ldots, r+g$) and has a probability of g/N to fall in

		S									
i	j	i_r-1	i_r	$\ldots$	i_r+g-1	i_r+g	$S[i]$	$S[j]$	$S[i]+S[j]$	$Output$
i_r-2	$i_r-1+g-A$	A	1	$\ldots$	B	C	/	/	/	/
i_r-1	i_r-1+g	B	1	$\ldots$	A	C	B	A	$B+A$	Z_1
i_r	i_r+g	B	C	$\ldots$	A	1	C	1	$C+1$	Z_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i_r+g-2	i_r-1-B	B	C	$\ldots$	A	1	/	/	/	/
i_r+g-1	i_r-1	A	C	$\ldots$	B	1	B	A	$B+A$	Z_1
i_r+g	i_r	A	1	$\ldots$	B	C	C	1	$C+1$	Z_2

Fig. 2. The Digraph Repetition Scenario

the i-affected area. Thus the probability of these two indices to remain in place is $(1-\frac{g}{N})^2 \cdot e^{-2g/N}$. For small g/N we can use the approximation $e^{-\varepsilon} \approx 1-\varepsilon$ to get $e^{-4g/N}$ and thus the overall probability of all indices to be in place is $e^{(8-8g)/N}$. □

Theorem 1. *For small values of G the probability of the pattern $AB\mathcal{S}AB$ in RC4 streams where S is a G-word string is $(1+e^{(-4-8G)/N}/N) \cdot 1/N^2$.*

Proof. Notice that g from Lemma 2 actually represents the shift between the digraphs and the real gap between the digraphs is $G = g-2$.

Let E_S (source event) be the event where the conditions of Lemma 2 are satisfied, i.e., $S_{r-1}[i_r] = 1$, $j_{r-1} = i_{r+G+1}$ and $j_{r+G+1} = i_{r-1}$ and let E_T (target event) be the event where there is equality between the output digraph of the origin pair and the end pair. The probability of E_S is N^{-3} and we use it to calculate the probability of E_T.

$$\begin{aligned}\mathbb{P}[E_T] &= \mathbb{P}[E_T|E_S] \cdot \mathbb{P}[E_S] + \mathbb{P}[E_T|\overline{E_S}] \cdot \mathbb{P}[\overline{E_S}] \\ &\geq e^{(-8-8G)/N}/N^3 + 1/N^2 \cdot (1-1/N^3) \\ &= 1/N^2 \cdot (1+e^{(-8-8G)/N}/N) + negl(1/N^2)\end{aligned}$$

□

The relative bias is $b_{DBL}(G) = \frac{e^{-8/N}}{N} \cdot (e^{-8/N})^G = C_1 \cdot C_2^G$ for the constants $C_2 = e^{-8/N} \approx 0.97$ and $C_1 = C_2/N$.

For the sake of simplicity the we made many heuristic assumptions during the analysis. Therefore, we carried out RC4 simulations to support the analysis and put the analytically calculated biases against the simulation result (see Figure 3)[2].

[2] In the simulation we used 2^{16} bitstreams of size 2^{24} each, ignoring cross-dependencies between the examined events and regarding every digraph with every gap as an independent event, meaning that every bitstream digraph was used for 64 experiments as the ending digraph and 64 experiments as the beginning digraph.

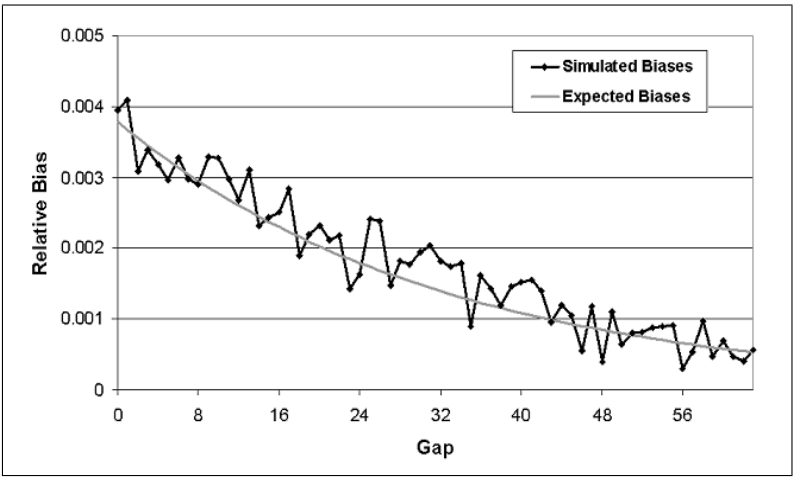

Fig. 3. Experimental vs. analytic results for biases of digraph repetitions

3.3 Digraph Repetition Distinguisher

In [4] Fluhrer and McGrew used information theoretic means to estimate the number of samples that are required for their distinguisher. They showed that this amount is inversely proportional to the discrimination between these distributions. Lemma 3 estimates the discrimination between two distributions that differ in a set of independent events.

Lemma 3. *Let $\mathcal{X}$ and $\mathcal{Y}$ be two distributions and suppose that the independent events $\{E_i : 1 \le i \le k\}$ occur with probabilities $p_{\mathcal{X}}(E_i) = p_i$ in $\mathcal{X}$ and $p_{\mathcal{Y}}(E_i) = (1 + b_i)p_i$ in $\mathcal{Y}$. Then the discrimination of the distributions is $\sum_i p_i b_i^2$.*

We prove this lemma in Appendix A.1.

Let us calculate this sum for the biases of digraphs repetitions with different gaps of at most M. The probability in random distributions is $1/N^2$ for all events and the relative bias in RC4 streams is $b_{DBL}(G)$.

$$\sum_{0 \le G \le M} p \cdot b_{DBL}^2(G) = p \cdot \sum_{0 \le G \le M} (C_1 \cdot C_2^G)^2 = p\, C_1^2 \cdot \sum_{0 \le G \le M} (C_2^2)^G$$

this is a partial sum of a geometric sequence and for $M \ge N/4$ the tail of the sum is very small

$$= p\, C_1^2 \cdot \frac{1 - C_2^{2M}}{1 - C_2^2} \approx N^{-4} \cdot \frac{e^{-16/N}}{1 - e^{-16/N}} \approx \frac{N^{-4}}{1 - (1 - 16/N)} = \frac{1}{16N^3}$$

The biases used by Fluhrer's distinguisher are in $7N$ events that has trivial probability $1/N^3$ and relative bias of $1/N$, and when summing them according to Lemma 3 we get an induced discrimination of $\sum pb^2 = 7/N^4$. Thus for a given success rate, a distinguisher based on our observations solely, requires

$102/N$ (less than half for $N = 256$) of the samples that are required by one that is based on Fluhrer's biases and a combined distinguisher requires $\frac{102}{N+102}$ (less than one third for $N = 256$) of this number.

Lemma 4. *The number of samples that is required for distinguishing two distributions that have discrimination D with success rate $1 - \alpha$ (for both directions) is $(1/D) \cdot (1 - 2\alpha) \cdot log_2 \frac{1-\alpha}{\alpha}$*

We prove this lemma in Appendix A.2. We use lemma 4 to estimate the numbers of samples that are required for different success rates, and get that less than 2^{29} samples are required for success rate of 90%, less than 2^{28} samples are required for success rate of 80% and less than 2^{26} samples are required for success rate of 2/3.

3.4 Applicability to RC4A

We divide the RC4 round into three parts

1. The *State update* part includes the changes in i and j and the swap of the corresponding values in S.
2. The *Output Index Calculation* part is the calculation of $S[i] + S[j]$
3. The *Emission* part is the selection of the output value from the permutation.

RC4A uses two instances of RC4 tables with cross references between them. Every round comprises two semi-rounds, in each of which an output word is emitted, and thus every round results with two output words. During the initialization stage both tables are initialized to pseudo-random permutations (we omit detailed description of this part) and during the generation stage both tables are involved in the generation of every output word; In the first semi-round the *state update* and the *output index calculation* are done in one table, whereas the *emission* part is done in the other table, namely the output word is taken from the other table (in the the calculated index). In the second semi-round, the roles of the tables change and the output is taken from the first table. Notice that on every such round both RC4A tables progress identically to a standard RC4 table and thus the state progression is similar to RC4. We will use this property to show that the statistics of digraphs repetition in RC4A is similar to its statistics in RC4.

Before getting to the fine details, let us give some intuition for why the digraph repetition patterns are biased also in RC4A. Most of the weaknesses of RC4 keystream generation rely on certain scenarios that are known to occur during RC4 progression. We refer to these scenarios as *special*. Most of the special scenarios rely on a small number of values that appear in certain locations of the RC4 table, which are used during the scenario. Scenarios that are based on a large number of values are less frequent and thus in order to maximize the probability of occurrence, a typical special scenario minimizes the number of starting conditions (known entries) by re-using the known entries for all three parts of the round; *State update* is crucial for assuring the correct progression of

the scenario. *Output index calculation* and *emission* are important for external expression of the scenario, sometimes resulting with biases in the keystream distribution. In RC4A the emission part is done in the second table and thus the same values cannot be used for both parts 1 and 3 or 2 and 3. Thus the expression of most of the special scenarios vanishes in RC4A and their induced weaknesses disappear.

However, the digraph repetition scenario is an exceptional one, since the permutation entries that guarantee the occurrence of the scenario are used only for *state update* and *output index calculation*. The expression of the scenario does not depend on any particular output value and thus the separation of the *emission* part does not affect the expression of the scenario. The output *indices* at close pairs of rounds are identical with high probability and the only requirement from the output values is to remain in the same locations during the intermediating rounds. This requirement is fully satisfied in RC4A since both tables evolve in the same rate as RC4 tables and therefore the digraph repetition scenario occurs in RC4A with similar statistics as in RC4.

The scenario in RC4A is as follows. We use permutations P_1 and P_2 and same symbols as in Figure 2. In addition, we ignore every second semi-round (which does not affect P_1).

1. Round t: The index i reaches the position before the value 1 in P_1 ($i_r - 1$). Output of the first semi-round is the value from P_2 in location $B + A$ (Z_1).
2. Round $t + 1$: The index i reaches the value of 1 in P_1 (i_r). Output of the first semi-round is the value from P_2 in location $C + 1$ (Z_2).
3. $g - 2$ rounds pass. Locations $B + A$ and $C + 1$ remain unchanged in P_2. Locations t, $t + 1$, $t + g$ and $t + g + 1$ remain unchanged in P_1.
4. Round $t + g$: The index i reaches again the position before the value 1 in P_1 ($i_r + g - 1$). Output of the first semi-round is the value from P_2 in location $B + A$ (again Z_1). In the second semi-round some arbitrary value is emitted.
5. Round $t + g + 1$: The index i reaches the value of 1 in P_1 ($i_r + g$). Output is the value from P_2 in location $C + 1$ (again Z_2).

Thus when omitting the even output words and concentrating in the sub-stream of odd words, or alternatively concentrating in sub-stream of even words) the same pattern $Z_1 Z_2 \underbrace{* \ldots *}_{g} Z_1 Z_2$ occurs with the same biased probability as in RC4 streams. When summing up the biases of these patterns in both sub-streams, we get a similar induced discrimination of $\frac{1}{16N^3}$.

Corollary 1. *There exists an algorithm that distinguishes RC4A streams from randomness with success probability of* $2/3$ *by analyzing less than* $2^{26.5}$ *samples.*

The difference between the distinguishers is due to the unavailability of Fluhrer's biases in RC4A. The best RC4A distinguisher that was described in the literature ([13]) requires 2^{59} output words, which is more than billion times the data for our distinguisher.

4 Recycling Attacks

4.1 Partial States and Fortuitous States

A significant part of published RC4 analysis is based on classification of RC4 states and partial states according to their cryptanalytic significance. We recall the definition of partial states and predictive states from [5].

Definition 1. *A d-state is a partially specified RC4 state, that includes i, j and d (not necessarily consecutive) elements of S.*

We use the notation $\{i_0, j_0, [x_0, x_1, \ldots, x_{d-1}]\}$ to specify the d-state $i = i_0$, $j = j_0$, $S[i_0 + 1, \ldots, i_0 + d] = [x_0, \ldots, x_{d-1}]$.

Definition 2. *Let D be a d-state and suppose that for a positive b there exist a sequence of b words that are output (in certain shifts) by every RC4 state that is compliant with D. Then D is said to be b-predictive.*

It was shown in [5] how b-predictive states can be used to attack RC4 by revealing parts of the internal state and constructing efficient distinguishing algorithms, in particular when $b = d$ (it was proved in [8] that $b \leq d$). A fortuitous state of order d is a d-predictive d-state in which the predicted d outputs are emitted immediately after the occurrence of the state. In Figure 4 we demonstrate predictiveness by presenting a family of $N - 1$ fortuitous states of order two, $\{x - 1, 0, [-1, x + 1]\}$ for every $x \neq 2$.

4.2 Recyclable States

A very interesting property of the scenario presented in Figure 4 is the fact that the swap on the second round reverts the swap from the first round, leaving the permutation in the same situation as it was at the beginning of the scenario. We formalize this behavior by defining *recyclable* states and then discuss how the cryptanalyst can exploit this property.

Definition 3. *For some d, let D be a d-state with $i = i^*$ and let I be the permutation consecutive interval that begins in $i^* + 1$, contains the d permutation entries of D and is the minimal interval that satisfies these requirements (thus $|I| >= d$). Suppose that in every D-compliant state, the permutation S after i leaves I satisfies again the permutation constraints of D. Then D is said to be recyclable.*

At the point where i leaves I, the permutation entries that are specified by D have the same values that they had when D occurred and they have a very

Round	i	j	$S[x]$	$S[x+1]$	$S[i]$	$S[j]$	$S[i]+S[j]$	*Output*
$x-1$	$x-1$	$x+2$	-1	$x+1$	/	/	/	/
x	x	$x+1$	$x+1$	-1	$x+1$	-1	x	$x+1$
$x+1$	$x+1$	x	-1	$x+1$	$x+1$	-1	x	-1

Fig. 4. A Family of Fortuitous States of order 2

high probability of more than e^{-d} to remain in place during the next $N - i$ rounds until the index i completes a full traversal of S. Combining in the $1/N$ probability of the index j to be also in place at that time, we get an overall probability of at least e^{-d}/N for another D-compliant state, exactly N rounds after the first one.

Moreover, D remains recyclable and the second occurrence of D causes a third occurrence with the same probability of e^{-d}/N and so forth. We formalize this observation in Theorem 2.

Theorem 2. *Let D be a recyclable RC4 d-state and suppose that the state of round t has the index i as specified by D and that the rest of the state is distributed uniformly. Then the probability of the states in rounds $t, t+N, t+2N, t+3N, \ldots, t+(k-1)N$ to be D-compliant is bounded from below by $e^{-d(k-1)}/N^{d+k}$.*

Proof. Since a D-compliant requires (except for i) d specific permutation elements and specific index j, the probability of the state in round t to be D-compliant is N^{-1-d}. The probability of round $t+N$ to have D-compliant state (given that round t had one) is at least e^{-d}/N and the same situation holds for round $t+2N, t+3N, \ldots, t+(k-1)N$ and thus the overall probability of the whole sequence to be D-compliant is at least $N^{-1-d} \cdot (e^{-d}/N)^{k-1} = e^{-d(k-1)}/N^{d+k}$. □

4.3 Availability of Recyclable Fortuitous States

We enumerated the fortuitous states of different orders to check whether and how many recyclable fortuitous states exist. The results are assorted where for some orders we found few states (9 states of order 3 and 100 states of order 5), whereas for other orders we found that most of the fortuitous states are recyclable. 512 out of 516 fortuitous states of order 2 are recyclable, more than 500 of them from the family $\{x, x+3, [-1, x+1 \text{ or } x+2]\}$. 4011 out of 6540 fortuitous states of order 4 are recyclable, more than 4000 of them from the family $\{x, x+6, [-3, 1, x+4 \text{ or } x+5 \text{ or } x+6 \text{ or } x+7, x \text{ or } x+1 \text{ or } x+2 \text{ or } x+3]\}$. There are no recyclable fortuitous states of order 6.

Due to time limitations, we were not able to complete the research on the availability of recyclable states of high order. Our intuition and some simulations we made lead us to the conjecture that recyclable fortuitous states of high order are rare or unavailable at all. However, when such states occur they tend to come in families, e.g., recyclable fortuitous states of an even order $2d$ tend to come in large families of about $N \cdot (2d)^d$ members (as described above for $d = 1, 2$), where the index i is "free" and the second half of the permutation constraints have some "freedom" (each entry has $2d$ alternatives).

4.4 Recycling Attacks

When the state in question is predictive, the recyclability property becomes extremely important since the occurrence of these states are expressed in the output stream.

Statistical Analysis of RC4 Streams

Theorem 3. *Let D be a recyclable b-predictive d-state and suppose that the predicted outputs of D are the values $z_0, \ldots, z_{b-1}$ in distances $r_0, \ldots, r_{b-1}$ after the occurrence of D. Then the following pattern has probability of at least $N^{-dk} \cdot (1 + \frac{N^{dk-d-k}}{e^{d(k-1)}})$.*

$$\underbrace{\overset{r_0}{z_0}, \ldots, \overset{r_1}{z_1}, \ldots, \overset{r_{b-1}}{z_{b-1}}, \ldots}_{N}, \underbrace{\overset{r_0+N}{z_0}, \ldots, \overset{r_{b-1+N}}{z_{b-1}}, \ldots}_{N}, \ldots, \underbrace{\overset{r_0+(k-1)N}{z_0}, \ldots, \overset{r_{b-1+(k-1)N}}{z_{b-1}}, \ldots}_{N}$$

Proof. Let E_T be the target event in which the output values of the dk rounds in question comply to the pattern. By Theorem 2 the probability for k occurrences of D in N-distances is $e^{-d(k-1)} \cdot N^{-d-k}$. We denote this event by E_S. When E_S occurs E_T occurs with probability 1 and otherwise E_T has the trivial probability of N^{-2k}.

$$\begin{aligned}
\mathbb{P}[E_T] &= \mathbb{P}[E_S] \cdot \mathbb{P}[E_T|E_S] + \mathbb{P}[\overline{E_S}] \cdot \mathbb{P}[E_T|\overline{E_S}] \\
&= e^{-d(k-1)} \cdot N^{-d-k} \cdot 1 + (1 - e^{-d(k-1)} \cdot N^{-d-k}) \cdot N^{-dk} \\
&= N^{-dk} \cdot (1 + \frac{N^{dk-d-k}}{e^{d(k-1)}}) - \underbrace{e^{-d(k-1)} \cdot N^{-d-(d+1)k})}_{negl(N^{-dk})}
\end{aligned}$$

□

Notice that for a fixed d the relative bias is proportional to $\left(\frac{N^{d-1}}{e^d}\right)^k$ and thus can increase to arbitrary values by increasing k. However, despite of the hugeness of these relative biases, they can be hardly used by distinguishing algorithms. When k grows, the original probability (N^{-2k}) decreases rapidly and the amount of data that is required for the distinguishing increases. However, large relative biases can be used for prediction of output bits and words with large advantages.

Theorem 4. *Let $\mathcal{X}$ be a distribution and suppose that a particular bit-pattern $B = b_0 b_1 \ldots b_{k-1}$ occurs with probability $p \cdot (1+b)$ in $\mathcal{X}$ (for $p = 2^{-k}$). In addition, suppose that for a parameter $v < k$ all the other k-bit strings with the same v-bit prefix as B have trivial probability of p. Then there exists an algorithm that once in $\frac{2^k}{2^{k-v}+b}$ samples (on average) of k-bit strings from $\mathcal{X}$ predicts the $(k-v)$-bit suffix with success probability $\frac{1+b}{2^{k-v}+b}$.*

We prove this theorem in Appendix B. Thus a k-bit pattern that occurs with relative bias b can be used to predict a bit (v=k−1) with success probability $\frac{1+b}{2+b} = 1/2 + \frac{b}{4+2b}$ once in every $\frac{2^k}{2+b}$ samples on average. The same pattern can be used to predict a byte (v=k−8) with success probability $\frac{1+b}{256+b}$ once in every $\frac{2^k}{256+b}$ samples on average.

We summarize the practical predictions that are derived from the combination of Theorems 3 and 4 in the table in Figure 5.

d	States	k	Relative Bias	$\mathbb{P}[E_T \mid \overline{E_S}]$	Samples Number	Byte Prediction	Bit Prediction
2	512	2	0.135	2^{-40}	$2^{29.9}$ (2^{23})	$1.135/N$	0.53
		3	4.7	2^{-56}	$2^{44.3}$ (2^{39})	$4.63/N$	0.85
		4	162	2^{-72}	$2^{55.7}$ $(2^{54.3})$	0.39	0.994
		5	5628	2^{-88}	$2^{66.5}$	0.956	≈ 1
3	9	2	12.75	2^{-56}	2^{49} $(2^{44.8})$	0.05	0.93
		3	41586	2^{-80}	$2^{61.5}$	≈ 1	≈ 1
4	4011	2	1200	2^{-72}	2^{50}	0.825	0.999
		3	$2^{28.5}$	2^{-104}	$2^{63.5}$	≈ 1	≈ 1

Fig. 5. Predictions of bits and bytes. The samples number are calculated for bit prediction and whenever the samples number for byte prediction is significantly different, we add it in parentheses

State Recovery Attack

Some of the state recovery attacks that are analyzed in [2] work very fast when more than 100 permutation elements are known. Let us present a way to obtain these 100 elements within relatively small effort using recyclable fortuitous states. A recyclable fortuitous state of order d repeats k times with probability $e^{-d(k-1)} \cdot N^{-d-k}$. The attacker can wait for the event where the expression pattern of this state repeats k times in shifts of N rounds ($e^{d(k-1)} \cdot N^{d+k}$) and guess that this external pattern stems from the corresponding internal pattern (with probability of almost 1). The amount of data that is required for this event to occur for appropriate selections of $k = d = 10$ is $e^{90}N^{20} \approx 2^{290}$. In that case, the attacker learns 100 permutation elements with probability 1. An equivalent attack that is based on occurrences of fortuitous states of order 100 get the hundred elements in a single state, but requires 2^{800} data in order to obtain them.

The attacks from [2] were tested in situations where the known elements are in a single permutation whereas in our case the known entries are actually k times the same d entries in k relatively close permutations. However, many situations are impossible under the constraints that the same values appear in the same entries during the kN rounds, e.g., the index j cannot touch these values (except for the fortuitous scenario), and the attacker can rule out many impossible branches and converge faster to the correct solution. Still the 2^{290} data complexity is a potential one, depending heavily of efficient ways to exploit these dk values. Another question that remains open at the moment regards the availability of recyclable fortuitous state of high order. There are such states of order 5, but there are none of order 6.

5 Summary

In this paper we presented new families of unique statistical patterns of the keystream generator of RC4. The families are large and the patterns are more frequent and more biased than previously known ones, and allow stronger attacks

and new attacks on RC4. These patterns were hidden for a long period mainly due to the fact that they are spread over distant rounds and thus could not be accidentally detected. However, the slow evolution of the permutation preserves many permutation elements with high probability along large number of rounds and we cannot rule out the possibility that this direction of research was not yet extracted.

References

1. Jovan Dj. Golić: Linear Statistical Weakness of Alleged RC4 Key-Stream Generator. EUROCRYPT: Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, EUROCRYPT'97 pp. 226–238
2. Lars R. Knudsen, Willi Meier, Bart Preneel, Vincent Rijmen and Sven Verdoolaege: Analysis Methods for (Alleged) RC4. ASIACRYPT: Advances in Cryptology, International Conference on the Theory and Applications of Cryptology and Information Security, ASIACRYPT'98 pp. 327–341
3. Serge Mister and Stafford E. Tavares: Cryptanalysis of RC4-like Ciphers. SAC: Selected Areas in Cryptography, SAC'98 pp. 131–143
4. Scott R. Fluhrer and David A. McGrew: Statistical Analysis of the Alleged RC4 Keystream Generator. FSE: Fast Software Encryption, FSE'00 pp. 19–30
5. Itsik Mantin and Adi Shamir: A Practical Attack on Broadcast RC4. FSE: Fast Software Encryption, FSE'01
6. Scott R. Fluhrer, Itsik Mantin and Adi Shamir: Weaknesses in the Key Scheduling Algorithm of RC4. SAC: Annual International Workshop on Selected Areas in Cryptography, SAC'01
7. I. Mironov: (Not So) Random Shuffles of RC4. Crypto'02
8. Souradyuti Paul and Bart Preneel: Analysis of Non-fortuitous Predictive States of the RC4 Keystream Generator. Progress in Cryptology - INDOCRYPT 2003: 4th International Conference on Cryptology in India INDOCRYPT'03 pp. 52–67
9. Souradyuti Paul and Bart Preneel: A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. Fast Software Encryption: 11th International Workshop, FSE'04 pp. 245–259
10. B. Zoltak: VMPC one-way function and stream cipher. Fast Software Encryption: 11th International Workshop, FSE'04 pp. 210–225
11. Jonathan J. Hoch and Adi Shamir: Fault Analysis of Stream Ciphers. CHES: Cryptographic Hardware and Embedded Systems, CHES'04 pp. 240–253
12. E. Biham, L. Granboulan and P. Nguyen: Impossible and Differential Fault Analysis of RC4. Fast Software Encryption: 12th International Workshop, FSE'05
13. Alexander Maximov: Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of the RC4 Family of Stream Ciphers. Fast Software Encryption: 12th International Workshop, FSE'05
14. Richard E. Blahut: Principles and Practice of Information Theory. Addisson-Wesley (1983)
15. Hal Finney: an RC4 cycle that can't happen. (1994)
16. Itsik Mantin: The Security of the Stream Cipher RC4. Master Thesis (2001) The Weizmann Institue of Science

17. Alexander L. Grosul and Dan S. Wallach: a Related-Key Cryptanalysis of RC4. Technical Report TR-00-358, Department of Computer Science, Rice University (2000)
18. Andrew Roos: A Class of Weak Keys in the RC4 Stream Cipher. Posted to sci.crypt (1995)

A Statistical Biases and Distinguishing Algorithms

A.1 Combining the Effect of Independent Events

We prove here Lemma 3. The discrimination of two distributions is given by the expression $\sum_s p_{\mathcal{X}} \lg \frac{p_{\mathcal{X}}(s)}{p_{\mathcal{Y}}(s)}$ where the sum is over all the possible strings. For a single event E with probabilities p and $p(1+b)$ the discrimination is (we use $q \stackrel{def}{=} 1-p$)

$$p \lg \frac{p}{p(1+b)} + q \lg \frac{q}{q-pb} = p \lg(1 - \frac{b}{1+b}) + q \lg(1 + \frac{pb}{q-pb})$$

We use the log approximation to get

$$\approx -p\frac{b}{1+b} + q\frac{pb}{q-pb} \approx pb^2$$

The following claim measures the effect of combining several distributions on the discrimination of the overall super-distribution.

Claim. Let $\mathcal{X}$ and $\mathcal{Y}$ be two distributions over the domain $\mathcal{S}$ and let $\mathcal{X}'$ and $\mathcal{Y}'$ be two distributions over the domain $\mathcal{S}'$. Let $\mathcal{X}\mathcal{X}'$ be the distribution (over $\mathcal{S} \times \mathcal{S}'$) that is created by concatenating the distributions $\mathcal{X}$ and $\mathcal{X}'$, and let $\mathcal{Y}\mathcal{Y}'$ be the equivalent distribution for $\mathcal{Y}$ and $\mathcal{Y}'$. Then $D(\mathcal{X}\mathcal{X}', \mathcal{Y}\mathcal{Y}') = D(\mathcal{X}, \mathcal{Y}) + D(\mathcal{X}', \mathcal{Y}')$.

Proof.

$$\begin{aligned}
D(\mathcal{X}\mathcal{X}', \mathcal{Y}\mathcal{Y}') &= \sum_{s\in\mathbf{S}, s'\in\mathbf{S}'} p_{\mathcal{X}\mathcal{X}'}(s,s') \lg \frac{p_{\mathcal{X}\mathcal{X}'}(s,s')}{p_{\mathcal{Y}\mathcal{Y}'}(s,s')} = \\
&= \sum_{s\in\mathbf{S}} \sum_{s'\in\mathbf{S}'} p_{\mathcal{X}}(s) p_{\mathcal{X}'}(s') \lg \frac{p_{\mathcal{X}}(s) p_{\mathcal{X}'}(s')}{p_{\mathcal{Y}}(s) p_{\mathcal{Y}'}(s')} = \\
&= \sum_{s\in\mathbf{S}} \sum_{s'\in\mathbf{S}'} p_{\mathcal{X}}(s) p_{\mathcal{X}'}(s') \left(\lg \frac{p_{\mathcal{X}}(s)}{p_{\mathcal{Y}}(s)} + \lg \frac{p_{\mathcal{X}'}(s')}{p_{\mathcal{Y}'}(s')} \right) = \\
&= \left(\sum_{s'\in\mathbf{S}'} p_{\mathcal{X}'}(s') \right) \cdot \left(\sum_{s\in\mathbf{S}} p_{\mathcal{X}}(s) \lg \frac{p_{\mathcal{X}}(s)}{p_{\mathcal{Y}}(s)} \right) + \\
&\quad + \left(\sum_{s\in\mathbf{S}} p_{\mathcal{X}}(s) \right) \cdot \left(\sum_{s'\in\mathbf{S}'} p_{\mathcal{X}'}(s') \lg \frac{p_{\mathcal{X}'}(s')}{p_{\mathcal{Y}'}(s')} \right) = \\
&= D(\mathcal{X}, \mathcal{Y}) + D(\mathcal{X}', \mathcal{Y}')
\end{aligned}$$

□

By applying Claim A.1 recursively we can get a generalization of this claim, saying that the overall discrimination of a combination of several independent distributions is the sum of the discriminations of these distributions. The combination of independent events for the same distribution (as described in Lemma 3) is equivalent to the combination of independent distributions and thus the discrimination can be summed over the events to get $\sum_i p_i b_i^2$.

A.2 Discriminations, Samples Numbers and Success Rates

We prove here Lemma 4. We denote the discrimination between two distributions by D and the discrimination between ℓ samples from these distributions by L. From [14], L is on one hand proportional to the number of samples and on the other hand induces a bound for the false positive rate α and the false negative rate β by the inequality

$$L = \ell D \geq \beta \lg \frac{\beta}{1-\alpha} + (1-\beta) \lg \frac{1-\beta}{\alpha}$$

For $\beta = \alpha$ we get $\ell \geq (1/D) \cdot (1-2\alpha) \cdot log \frac{1-\alpha}{\alpha}$.

B Statistical Biases and Bit Predictions

We prove here Theorem 4.

Proof. Let E_{PREF} and E_{SUFF} be the events where the v-bit prefix of B and the $(k-v)$-bit suffix of B occur, and let E be the joint event where B occur (namely, $E = E_{PREF} \wedge E_{SUFF}$). Let $E' = E_{PREF} \wedge \overline{E_{SUFF}}$. The algorithm analyzes the output stream of $\mathcal{X}$ and locates occurrences of E_{PREF}. Notice that the bias of E affects the probability E_{PREF} in the following manner:

$$\mathbb{P}[E_{PREF}] = \mathbb{P}[E] + \mathbb{P}[E'] = p(1+b) + (2^{k-v} - 1)p = \frac{2^{k-v} + b}{2^k}$$

In such occurrences the algorithm predicts that E_{SUFF} will occur with the following success probability

$$\begin{aligned} \mathbb{P}[E_{SUFF} | E_{PREF}] &= \frac{\mathbb{P}[E_{SUFF} \cap E_{PREF}]}{\mathbb{P}[E_{PREF}]} = \\ &= \frac{\mathbb{P}[E]}{2^{-v}(1 + 2^{v-k}b)} = \\ &= \frac{2^{-k}(1+b)}{2^{-v}(1+2^{v-k}b)} = \frac{1+b}{2^{k-v}+b} \end{aligned}$$

□

Related-Key Boomerang and Rectangle Attacks

Eli Biham[1], Orr Dunkelman[1,⋆], and Nathan Keller[2]

[1]Computer Science Department, Technion,
Haifa 32000, Israel
{biham, orrd}@cs.technion.ac.il
[2]Einstein Institute of Mathematics, Hebrew University,
Jerusalem 91904, Israel
nkeller@math.huji.ac.il

Abstract. The boomerang attack and the rectangle attack are two attacks that utilize differential cryptanalysis in a larger construction. Both attacks treat the cipher as a cascade of two sub-ciphers, where there exists a good differential for each sub-cipher, but not for the entire cipher. In this paper we combine the boomerang (and the rectangle) attack with related-key differentials.

The new combination is applicable to many ciphers, and we demonstrate its strength by introducing attacks on reduced-round versions of AES and IDEA. The attack on 192-bit key 9-round AES uses 256 different related keys. The 6.5-round attack on IDEA uses four related keys (and has time complexity of $2^{88.1}$ encryptions). We also apply these techniques to COCONUT98 to obtain a distinguisher that requires only four related-key adaptive chosen plaintexts and ciphertexts. For these ciphers, our results attack larger number of rounds or have smaller complexities then all previously known attacks.

1 Introduction

The *boomerang attack* [23] is an adaptive chosen plaintext and ciphertext attack utilizing differential cryptanalysis [6]. The cipher is treated as a cascade of two sub-ciphers, where a short differential is used in each of these sub-ciphers. These two differentials are combined in an elegant way to suggest an adaptive chosen plaintext and ciphertext property of the cipher that has high probability.

The boomerang attack was further developed in [18] into a chosen plaintext attack called the *amplified boomerang attack*. The transformation uses birthday-paradox techniques to eliminate the adaptive nature of the attack, by encrypting large sets of plaintexts with the required input difference. After the encryption of the plaintext pairs, the attacker searches for quartets of plaintexts that satisfy the same conditions as if these quartets were constructed in the boomerang process. The transformation to a chosen plaintext attack (instead of an adaptive

⋆ The research presented in this paper was supported by the Clore scholarship programme.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 507–525, 2005.

chosen plaintexts and ciphertexts attack) has price both in a much larger data complexity and a much more complicated algorithm for the identification of the right quartets. After its introduction, the amplified boomerang attack was further developed into the rectangle attack [4]. The rectangle attack uses a more careful analysis that shows that the probability of a right quartet is significantly higher than suggested by the amplified boomerang attack. An optimized algorithm for finding and identifying the right rectangle quartets was given in [5].

Related-key attacks [1] consider the information that can be extracted from two encryptions using related keys. The concept was used in [19] to present the idea of related-key differentials. These differentials study the development of differences in two encryptions under two related keys.

In this paper we show how to combine these attacks with related-key differentials. In [20], a boomerang attack that uses one regular differential along with one related-key differential is introduced. Both this paper and [16] independently developed the idea of using two related-key differentials, one for each sub-cipher, simultaneously. The major difference between this work and [16] is the idea of using more than one key difference in the differentials to obtain much better attacks.

The basic related-key boomerang attack (which is similar to the one presented in [16]) is aimed against ciphers whose subkeys are linear functions of the key. In this case, a fixed key difference yields a known subkey differences.

The more complicated version of the attack deals with ciphers whose subkeys are not linear functions of the keys. In this case, the attacker has to take into consideration the fact that the initial key difference does not guarantee the subkey differences used in the differential. In order to overcome this problem, we use differential properties of the key schedule algorithm and use several pairs of keys. This leads to the introduction of structures of keys under which structures of plaintexts are being encrypted or decrypted.

We take advantage of the fact that in boomerang and rectangle attacks the used differentials are shorter, and thus the diffusion of differences in the subkeys can be used better than in ordinary related-key differential case.

Finally, we apply our attack against several block ciphers:AES [12], IDEA [21], and COCONUT98 [22]. The attack on 9-round AES-192 requires 2^{87} related-key chosen plaintexts (2^{79} plaintexts encrypted under 256 different keys), and has running time of 2^{125} encryption. The attack on 6.5-round IDEA requires $2^{59.8}$ related-key chosen plaintexts ($2^{57.8}$ plaintexts encrypted under four keys), and has time complexity of $2^{88.1}$ encryptions. We also apply these techniques to COCONUT98 to obtain a distinguisher that requires only four related-key adaptive chosen plaintexts and ciphertexts encrypted under two different keys. We summarize our results along with previously known results on the respective ciphers in Table 1.

This paper is organized as follows: In Section 2 we give a brief description of the boomerang and the rectangle attacks. In Section 3 we describe the new related-key boomerang and rectangle attacks. In Section 4 we present a related-key rectangle attack on 9-round AES-192 and 10-round AES-256. In Section 5

Table 1. Summary of the Previous Attacks and of Our New Attacks

Cipher	Number of Rounds	Complexity Data	Complexity Time	Number of Source Keys	
AES-192	8	$2^{128} - 2^{119}$ CP	2^{188}	1	[14]
(12 rounds)	8	2^{89} RK-CP	2^{183}	2	[17]
	8	$2^{86.5}$ RK-CP	$2^{86.5}$	4	[16]
	9	2^{86} RK-CP	2^{125}	256	Section 4
AES-256	8	$2^{128} - 2^{119}$ CP	2^{204}	1	[14]
(14 rounds)	9	2^{85} RK-CP	$5 \cdot 2^{224}$	256	[14]
	10	$2^{114.9}$ RK-CP	$2^{171.8}$	256	Section 4
COCONUT98	full	2^{16} ACPC	2^{38}	1	[23]
	full†	4 RK-ACPC	1	2	Section 5
IDEA	5	2^{24} CP	2^{126}	1	[13]
(8.5 rounds)	5.5†	$2^{51.6}$ RK-ACPC	1	4	Section 6
	6	$2^{51.6}$ RK-ACPC	2^{48}	4	Section 6
	6.5	$2^{59.8}$ RK-CP	$2^{88.1}$	4	Section 6

† – Distinguishing attack, RK – Related-key, CP – Chosen plaintext, ACPC – Adaptive chosen plaintext and ciphertext
Time complexity is measured in encryption units

we present a related-key boomerang distinguisher for COCONUT98. Section 6 describes our results on IDEA. Finally, Section 7 summarizes this paper.

2 Boomerang and Rectangle Attacks

The main idea behind the boomerang attack [23] is to use two short differentials with high probabilities instead of one long differential with a low probability. The motivation for such an attack is quite apparent, as in many block ciphers it is easier to find short differential with high probability than to find a long differential with high enough probability (or even impossible).

We assume that a block cipher $E: \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$ can be described as a cascade, i.e., $E = E_1 \circ E_0$, such that for E_0 there exists a differential $\alpha \rightarrow \beta$ with probability p, and for E_1 there exists a differential $\gamma \rightarrow \delta$ with probability q. The distinguisher is the following boomerang process:

- Ask for the encryption of a pair of plaintexts (P_1, P_2) such that $P_1 \oplus P_2 = \alpha$, and denote the corresponding ciphertexts by (C_1, C_2).
- Calculate $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$, and ask for the decryption of the pair (C_3, C_4). Denote the corresponding plaintexts by (P_3, P_4).
- Check whether $P_3 \oplus P_4 = \alpha$.

The boomerang attack uses the first differential $(\alpha \rightarrow \beta)$ for E_0 with respect to the pairs (P_1, P_2) and (P_3, P_4), and uses the second differential $(\gamma \rightarrow \delta)$ for E_1 with respect to the pairs (C_1, C_3) and (C_2, C_4). The first differential is used in the backward direction for the pairs (P_3, P_4), and the second differential is used in the backward direction for both respective pairs.

For a random permutation the probability that the last condition is satisfied is 2^{-n}. For E, the probability that the pair (P_1, P_2) is a right pair with respect to the first differential $(\alpha \to \beta)$ is p. The probability that both pairs (C_1, C_3) and (C_2, C_4) are right pairs with respect to the second differential is q^2. If all these are right pairs, then $E_1^{-1}(C_3) \oplus E_1^{-1}(C_4) = \beta = E_0(P_3) \oplus E_0(P_4)$, and thus with probability p, $P_3 \oplus P_4 = \alpha$. The total probability of this quartet of plaintexts and ciphertexts to satisfy the boomerang conditions is $(pq)^2$.

The attack can be mounted for all possible β's and γ's simultaneously (as long as $\beta \neq \gamma$). Thus, a right quartet for E is encountered with probability no less than $(\hat{p}\hat{q})^2$, where:

$$\hat{p} = \sqrt{\sum_{\beta} \Pr^2[\alpha \to \beta]}, \qquad \text{and} \qquad \hat{q} = \sqrt{\sum_{\gamma} \Pr^2[\gamma \to \delta]}.$$

For the complete analysis of the boomerang attack see [23].

As the boomerang attack requires adaptive chosen plaintexts and ciphertexts, many of the techniques that were developed for using distinguishers in key recovery attacks cannot be applied. This led to the introduction of a chosen plaintext variant of the boomerang attack called the *amplified boomerang attack* [18]. The key idea behind the transformation is to encrypt many plaintext pairs with input difference α, and to look for quartets that conform to the requirements of the boomerang process.

This kind of transformation is common, and can be achieved by birthday-paradox arguments. A more careful analysis shows that two pairs, $(P_1, P_2 = P_1 \oplus \alpha)$ and $(P_3, P_4 = P_3 \oplus \alpha)$, form a right quartet if three conditions are satisfied:

1. $E_0(P_1) \oplus E_0(P_2) = \beta = E_0(P_3) \oplus E_0(P_4)$.
2. $E_0(P_1) \oplus E_0(P_3) = \gamma$ (which leads to $E_0(P_2) \oplus E_0(P_4) = \gamma$ if this condition and the previous one hold).
3. $C_1 \oplus C_3 = \delta = C_2 \oplus C_4$.

The usual assumptions are that each of these conditions is independent of the rest, and that the probability that a quartet would become a right quartet is $p^2 \cdot 2^{-n} \cdot q^2$. We note that if the conditions are dependent on each other, refined algorithms may use these relations for achieving higher probabilities. The low probability follows from the fact that the event $E_0(P_1) \oplus E_0(P_3) = \gamma$ occurs with probability of 2^{-n}. The analysis in [18] shows that out of N plaintext pairs, the number of right quartets is expected to be $N^2 2^{-(n+1)} p^2 q^2$.

Besides the lower probabilities, the transformation into a chosen plaintext attack introduces the problem of identifying the right quartets. In the boomerang attack the pair (P_3, P_4) that we test is known. In the amplified boomerang attack, this is not the case. Instead, the attacker has to search for the right quartets among all possible quartets.

The rectangle attack [4] shows that it is possible to use all the possible β's and γ's simultaneously, and presents additional improvements over the amplified boomerang attack. These improvements increase the probability of a quartet to

be a right quartet, and N plaintext pairs with input difference α are expected to produce $N^2 2^{-n} \hat{p}^2 \hat{q}^2$ right quartets[1], where $\hat{p}$ and $\hat{q}$ are as defined above. In [5] an optimized method of finding the right rectangle quartets is presented.

3 Related-Key Boomerang and Rectangle Attacks

A regular differential deals with some plaintext difference ΔP and a ciphertext difference ΔC such that

$$\Pr_{P,K}[E_K(P) \oplus E_K(P \oplus \Delta P) = \Delta C]$$

is high enough (or zero [3]). The common assumption is that this probability is quite uniform over all keys and plaintexts. If this is not the case, a weak key class can be found, i.e., a set of keys for which the above probability is far from average (either very high or very low).

A related-key differential is a triplet of a plaintext difference ΔP, a ciphertext difference ΔC, and a key difference ΔK, such that

$$\Pr_{P,K}[E_K(P) \oplus E_{K \oplus \Delta K}(P \oplus \Delta P) = \Delta C]$$

is high enough (or zero). Again, there is an assumption that this probability is independent of P and K. Sometimes the relation between the keys is more complex than XOR with some constant ΔK (see [1, 9]), but for sake of simplicity we shall deal only with this kind of relation in this paper, even though our technique is not restricted for this case.

3.1 Related-Key Boomerang Attacks

Let us assume that we have a related-key differential $\alpha \rightarrow \beta$ of E_0 under a key difference ΔK_0 with probability p. Assume also that we have another related-key differential $\gamma \rightarrow \delta$ for E_1 under key difference ΔK_1 with probability q.

The related-key boomerang process involves four different unknown (but related) keys — K_a, $K_b = K_a \oplus \Delta K_0$, $K_c = K_a \oplus \Delta K_1$, and $K_d = K_a \oplus \Delta K_0 \oplus \Delta K_1$. The attack is performed by the following algorithm:

- Choose a plaintext P_a at random and compute $P_b = P_a \oplus \alpha$.
- Ask for the encryption of P_x under K_x, i.e., $C_a = E_{K_a}(P_a)$ and $C_b = E_{K_b}(P_b)$.
- Compute $C_c = C_a \oplus \delta$ and $C_d = C_b \oplus \delta$.
- Ask for the decryption of C_x under K_x, i.e., $P_c = E^{-1}_{K_c}(C_c)$ and $P_d = E^{-1}_{K_d}(C_d)$.
- Test whether $P_c \oplus P_d = \alpha$.

[1] This number is a lower bound for the expected number. For the complete analysis see [4].

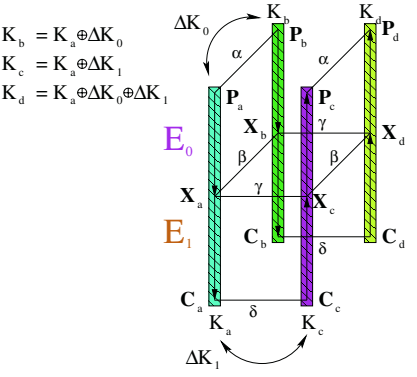

Fig. 1. A Related-Key Boomerang Quartet

It is easy to see that for a random permutation, the probability that the last condition is satisfied is 2^{-n}. For E the probability that this condition is satisfied is p^2q^2 just like for a regular boomerang attack. Figure 1 outlines a quartet satisfying all the required conditions.

The attack can use multiple differentials for E_0 and E_1 (just like in a regular boomerang attack), under the strict condition that all related-key differentials used in E_0 have the same key difference ΔK_0 and the same input difference α, and that all related-key differentials used in E_1 have the same key difference ΔK_1 and the same output difference δ. Thus, the probability of a quartet to be a right quartet is $\hat{p}^2\hat{q}^2$.

When the key schedule algorithm is linear then given a key difference all subkey differences are known, and are easily predicted. In this case the attack algorithm from [5] can be adapted. Otherwise, if the key schedule algorithm is non-linear, the exact key difference needed to satisfy the subkey differences of the related-key differential might be unknown. In the latter case, the attacker examines the differential properties of the key schedule algorithm and computes the probability that a given key differences evolves into the required subkey differences. Then, the attacker repeats the attack with various key differences, such that in one (or more) of the cases, the key difference causes the subkey differences needed for the related-key differential. Note that this attack is actually a multiple application of the basic related-key boomerang/rectangle attacks. An example of such an attack is the attack on AES-192 presented in Section 4.

3.2 Related-Key Rectangle Attack

The transformation of the related-key boomerang attack into a related-key rectangle attack is similar to the transformation of the boomerang attack into the rectangle attack. Assume that E can be decomposed as before, where $\alpha, \delta, \hat{p}$, and $\hat{q}$ have the same meaning. Then, related-key rectangle distinguisher is as follows:

– Choose N plaintext pairs $(P_a, P_b = P_a \oplus \alpha)$ at random and ask for the encryption of P_a under K_a and of P_b under K_b.

- Choose N plaintext pairs $(P_c, P_d = P_c \oplus \alpha)$ at random and ask for the encryption of P_c under K_c and of P_d under K_d.
- Search for quartets of plaintexts (P_a, P_b, P_c, P_d) and the corresponding ciphertexts (C_a, C_b, C_c, C_d), satisfying $C_a \oplus C_c = C_b \oplus C_d = \delta$.

The analysis of the related-key rectangle attack is similar to the analysis of the rectangle attack (with the same modifications that were presented at the related-key boomerang attack). Starting with N plaintext pairs with input difference α to be encrypted under K_a and K_b, we expect $N^2 2^{-n} (\hat{p}\hat{q})^2$ right quartets. We note that under the requirement that we encrypt distinct values N is no longer bounded by 2^{n-1} (as in the rectangle attack), but rather can be up to 2^n pairs in most of the cases (when $\Delta K_0 = 0, \Delta K_1 = 0$, or $\Delta K_0 = \Delta K_1$ the value of 2^{n-1} is still the bound).

The step of finding the right quartets is technical in nature (the simplest algorithm is trying all possible quartets, which is very inefficient). When the key difference predicts the required subkey differences with probability 1, then the attack algorithm of [5] can be easily adapted. Otherwise, we have to perform a method similar to the one of the boomerang attack — repeat the attack for several quartets of keys.

4 Related Key Rectangle Attacks on AES-192

The advanced encryption standard [12] is an SP-network that supports key sizes of 128, 192, and 256 bits. The 128-bit plaintexts are treated as byte matrices of size 4x4, where each byte represents a value in $GF(2^8)$. An AES round applies four operations to the state matrix: SubBytes (SB) – applying the same S-box 16 times in parallel on each byte of the state, ShiftRows (SR) – cyclic shift of each row (the i'th row is shifted by i bytes to the right), MixColumns (MC) – multiplication of each column by a constant 4x4 matrix over the field $GF(2^8)$, and AddRoundKey (ARK) – XORing the state and a 128-bit subkey.

The MixColumns operation is omitted in the last round, and an additional AddRoundKey operation is performed before the first round (a whitening key). We denote the subkey of round i by superscript K^{i+1}, i.e., the first (whitening) key is K^0, the subkey of the first round is K^1, etc. We also denote the byte in the i'th row and the j'th column of the state matrix by byte $j * 4 + i$, where $i, j \in \{0, 1, 2, 3\}$.

The number of rounds depends on the key length: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. The rounds are numbered $0, \ldots, Nr - 1$, where Nr is the number of rounds ($Nr \in \{10, 12, 14\}$). For sake of simplicity we shall denote AES with n-bit keys by AES-n, i.e., AES with 128-bit keys (and thus with 10 rounds) is denoted by AES-128.

The best published differential-based attack on AES is a boomerang attack on a 6-round reduced version of AES-128 [7]. The attack requires 2^{71} adaptive chosen plaintexts and ciphertexts, and its time complexity is equivalent to 2^{71} encryptions. The best known attack on AES-192 is a SQUARE attack on 8

rounds [14]. The attack requires almost the entire code book ($2^{128} - 2^{119}$ chosen plaintexts) and has a time complexity equivalent to 2^{188} encryptions. The best related-key attack against AES-192 is a related-key rectangle attack on 8 rounds [16]. It requires $2^{86.5}$ chosen plaintexts (encrypted under four keys) and has a time complexity equivalent to $2^{86.5}$ encryptions.

In this section we present a related-key rectangle attack on 9-round AES-192. The attack has data complexity of 2^{87} related-key chosen plaintexts (2^{79} chosen plaintexts encrypted under 256 keys), and time complexity of 2^{125} encryptions. By using similar techniques, one can attack 10-round AES-256 using $2^{114.9}$ related-key chosen plaintexts ($2^{106.9}$ chosen plaintexts encrypted under 256 keys) with time complexity of $2^{171.8}$ encryptions.

We concentrate on AES-192, as it demonstrates our attack when the key schedule is not linear, but is still very close to linear. The attack uses structures of plaintexts, encrypted under structures of keys, where the structures of keys are sets of keys selected to assure that there exists a quartet of keys whose subkeys satisfy the required differences.

4.1 Preliminaries for the Attack on 9-Round AES-192

The application of the related-key rectangle attack to AES-192 is as follows: We start by finding two good related-key differentials. In the second differential the key difference cannot guarantee the required subkey differences needed for the differential. Thus, we repeat the attack with 127 key differences, as we are assured that for at least one of these values the subkey differences are satisfied.

We decompose 9-round AES-192 (starting in the third round — rounds 2–10) as follows: rounds 2 and 3 are the rounds before the distinguisher, rounds 4–6 are E_0 (without the key addition of round 6), rounds 7–9 (with the key addition of round 6) are E_1, and round 10 is the round after the distinguisher.

Let θ_0 be a fixed 8-bit known non-zero difference, and let $\theta = (\theta_0, 0, 0, 0)$ be a 32-bit difference (the three 0's are byte differences). Let $\chi = (\chi_0, \chi_1, \chi_2, \chi_3) = MC(\theta)$, where χ_i are non-zero byte differences, and MC is the MixColumns operation. The value of χ is known as the MC operation is linear, thus, differentials propagate linearly through it as well.

Let θ by an input difference to the $MC(BS())$ operations. We denote all 127 possible output differences by MB, i.e., $MB = \{MC(BS(x)) \oplus MC(BS(x \oplus \theta))$.

4.2 The First Differential (E_0)

The first differential (for rounds 4–6) is as follows: the subkey difference of K^4 is equal to the input difference and being of the form $\alpha = \Delta K^4 = (0, 0, \chi, \chi)$ (here the 0's are 32-bit differences). After the key addition, the difference of the data is zero, which remains through round 4 with probability 1. We set $\Delta K^5 = 0$, and we get that the zero difference remains after round 5 with probability 1 as well. Then, the subkey difference ΔK^6 is necessarily $(\chi, 0, 0, 0)$, which leads to the activation of four S-boxes at round 6. As χ_i are all known and fixed, there are 127 possible output differences in each of the four active S-boxes, of which

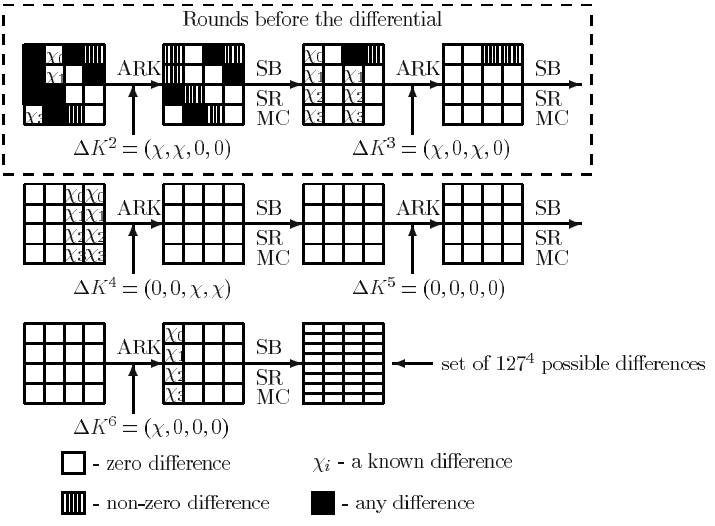

Fig. 2. The First Differential Used in the Attack (and the Two Preceding Rounds)

one occurs with probability 2^{-6}, and the rest with probability 2^{-7}. Therefore, the probabilities of the 127^4 β output differences are distributed as follows: one has probability 2^{-24}, $4 \cdot 126$ have probability 2^{-25}, and so forth, up to 126^4 with probability 2^{-28}. As we use all these differentials simultaneously, we get that $\hat{p} = \sqrt{\sum_\beta \Pr^2[\alpha \to \beta]} = 2^{-13.96}$.

We note that the above describes a differential characteristic (we predict the development of the difference in all rounds). In the case of the AES, the probability that $\alpha \to \beta$ is very close to the probability of the characteristic.

We look for the (related-key) input difference to round 2 that leads to an α difference at the input of round 4. The input difference for round 2 consists of four S-boxes with a zero input difference (bytes $9, 10, 14, 15$), three S-boxes whose non-zero difference is known (bytes $3, 4, 5$), two additional S-boxes with unknown non-zero difference (bytes $11, 12$), and the remaining seven S-boxes can have any difference. We denote the difference in the bytes whose difference is known by ΔM_0, where we put zeroes in the bytes whose difference in unknown (ΔM_0 has four non-zero bytes). We outline these differences of the differential and the preceding rounds in Figure 2. The first differential's key difference is $\Delta K_0 = (\chi, 0, 0, 0, \chi, 0)$, and we outline the subkey differences in Table 2.

4.3 The Second Differential (E_1)

The second differential predicts the differences in E_1 (rounds 7–9). The input difference γ equals the subkey difference, and both are $\gamma = \Delta K^7 = (0, 0, \theta, \theta)$. Thus, after the key addition the difference is zero, which passes with probability 1 through round 7. We take $\Delta K^8 = 0$, and thus, the zero difference remains with

Table 2. Subkey Differences for the Key Difference $\Delta K_0 = (\chi, 0, 0, 0, \chi, 0)$ (The subkey differences of the differential are in bold)

Subkey	Difference	Subkey	Difference	Subkey	Difference	Subkey	Difference
K^0	$(\chi, 0, 0, 0)$	K^2	$(\chi, \chi, \mathbf{0}, \mathbf{0})$	K^4	$(\mathbf{0}, \mathbf{0}, \chi, \chi)$	K^6	$(\chi, \mathbf{0}, \mathbf{0}, \mathbf{0})$
K^1	$(\chi, 0, \chi, \chi)$	K^3	$(\chi, \mathbf{0}, \chi, \mathbf{0})$	K^5	$(\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$		

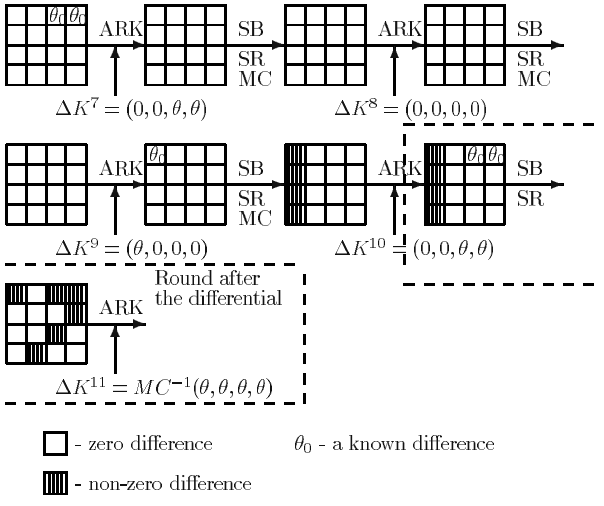

Fig. 3. The Second Differential Used in the Attack (and the Following Round)

probability 1 also after round 8. Once we select ΔK^7 and ΔK^8, then necessarily $\Delta K^9 = (\theta, 0, 0, 0)$. Thus, the key mixing before round 9 introduces a difference θ_0 in byte 0. This byte difference creates a difference in a full column before the addition of K^{10}. The subkey difference is $\Delta K^{10} = (0, 0, \theta, \theta)$, hence, the output difference δ of the differential has four active bytes (in bytes $0, \ldots, 3$) and twelve bytes with a zero difference. The difference in bytes $0, \ldots, 3$ is unknown but is restricted to a set of 127 possible differences. We outline the differences of this differential and the following round in Figure 3. We note that the differential has probability 1, leading to $\hat{q} = 1$.

The subkey differences of the second differential are given in Table 3. The key difference that achieves the required subkey differences for the second differential is of the form $\Delta K_1 = (\mu, \theta \oplus \mu, \theta, 0, \theta, 0)$, where $\mu = (0, \mu_1, 0, 0)$ and $\theta_0 \rightarrow \mu_1$ by the S-box. The exact value of μ_1 is unknown, but μ_1 must be one of 127 possible values.

4.4 The Structure of Keys

Let K_a be the unknown key which we would like to recover. The related-key that is required for the first differential is $K_b = K_a \oplus \Delta K_0$.

Table 3. Subkey Differences for the Key Difference $\Delta K_1 = (\mu, \theta \oplus \mu, \theta, 0, \theta, 0)$ (The subkey differences in the rounds of the attack are in bold)

Subkey	Difference	Subkey	Difference	Subkey	Difference	Subkey	Difference
K^0	$(\mu, \theta \oplus \mu, \theta, 0)$	K^3	$(\theta, 0, 0, 0)$	K^6	$(\theta, 0, \theta, 0)$	K^9	$(\theta, \mathbf{0}, \mathbf{0}, \mathbf{0})$
K^1	$(\theta, 0, \mu, \theta)$	K^4	$(\theta, 0, \theta, \theta)$	K^7	$(\mathbf{0}, \mathbf{0}, \theta, \theta)$	K^{10}	$(\mathbf{0}, \mathbf{0}, \theta, \theta)$
K^2	$(0, 0, \theta, \theta)$	K^5	$(\theta, \theta, 0, 0)$	K^8	$(\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$	K^{11}	$(\theta, \theta, \theta, \theta)$

Table 4. The Keys Required for the Related-Key Rectangle Attack

Key	Values	#	Key	Values	#
K_a	K_a	1	K_c	$\{K_a \oplus (\mu, \theta \oplus \mu, \theta, 0, \theta, 0)\}$	127
K_b	$K_a \oplus (\chi, 0, 0, 0, \chi, 0)$	1	K_d	$\{K_a \oplus (\mu \oplus \chi, \theta \oplus \mu, \theta, 0, \theta \oplus \chi, 0)\}$	127

Examine K_a and the subkey differences needed for the second differential. There are 127 possible related keys with which K_a may have the required subkey differences. Denote this set of keys by KS_c, which is actually all the keys that satisfy $K_a \oplus \Delta K_1$ for some value of μ_1, where $\mu = (0, \mu_1, 0, 0)$ and $\Delta K_1 = (\mu, \theta \oplus \mu, \theta, 0, \theta, 0)$. One key of this set satisfies the required subkey differences with respect to K_a, and we denote it by K_c.

We denote the key with difference ΔK_0 from K_c by K_d, i.e., $K_d = K_c \oplus \Delta K_0$. If we want to use the four keys, it must hold that K_b and K_d have the subkey difference required by the second differential, i.e., $K_b = K_d \oplus \Delta K_1$. In this case this is true, as there is no difference between K_a and K_b in the word that we need the equality (for the S-box application). Thus, $K_a \oplus K_c = K_b \oplus K_d$ is true. We outline the sets of keys in Table 4.

Note that we can choose a smaller number of keys, such that using the birthday-paradox we get with high probability the required quartets of keys, but then the attack may fail in a small fraction of the cases.

4.5 The Attack

The main idea of the attack is to try all possible quartets of keys (K_a, K_b, K_c, K_d) that can satisfy the required subkey differences, by performing the rectangle attack from [5] on each of these possibilities. The attack presented here is an optimized version of this idea, in which we take advantage of the fact that once the keys with the right relations are encountered, then there is no need to continue the attack for other quartets of keys. The attack algorithm is as follows:

1. Data Generation:
   (a) Generate 64 structures $S_1^a, \ldots, S_{64}^a$ of 2^{72} plaintexts each, where in each structure the 56 bits of bytes $3, 4, 5, 9, 10, 14, 15$ are fixed. Ask for the encryption of the structures under K_a.

(b) XOR any plaintext encrypted under K_a with ΔM_0, and ask for the encryption of the resulting plaintext under K_b (to obtain $S_1^b, \ldots, S_{64}^b$).
(c) For any possible value μ_1 such that $\theta_0 \rightarrow \mu_1$, let $\Delta K_1 = (\mu, \theta \oplus \mu, \theta, 0, \theta, 0)$, perform:
   i. Generate 64 structures $S_1^{c'}, \ldots, S_{64}^{c'}$ of 2^{72} plaintexts each, where in each structure the 56 bits of bytes $3, 4, 5, 9, 10, 14, 15$ are fixed. Ask for the encryption of the structure under $K_{c'} = K_a \oplus \Delta K_1$.
   ii. XOR any plaintext encrypted under $K_{c'}$ with ΔM_0, and ask for the encryption of the resulting plaintexts under $K_{d'} = K_{c'} \oplus \Delta K_0$ (to obtain $S_1^{d'}, \ldots, S_{64}^{d'}$).

2. Data Analysis: For any $K_{c'}$, the respective $K_{d'}$, and the corresponding structures:
   (a) For any pair of structures $S_i^b, S_j^{d'}$ perform:
      i. Insert the 2^{72} ciphertexts of $S_i^b, S_j^{d'}$ into a hash table indexed by the 80 bits of bytes $4, 5, 6, 7, 9, 10, 11, 13, 14, 15$. About $2^{72} \cdot 2^{72}/2^{80} = 2^{64}$ collisions are expected.
      ii. For each 80-bit collision (where one ciphertext is from S_i^b and one is from $S_j^{d'}$) check that the ciphertext difference in bytes 2 and 3 may be caused by an input difference θ_0 to the S-box. If this is not the case, discard the pair.
      iii. For each of the expected 2^{62} remaining pairs, try all 2^{32} possible values of bytes $0, 7, 10, 13$ of K^{11}, and partially decrypt the pair. If the difference of the partially decrypted pair is in MB, add the pair to a list of pairs related to the subkey. We note that each pair is expected to be in 127 lists, and that each list contains about $127 \cdot 2^{30} \approx 2^{37}$ pairs.
   (b) Insert **all** the ciphertexts of $S_1^a, \ldots, S_{64}^a, S_1^{c'}, \ldots, S_{64}^{c'}$ into a hash table indexed by the 80 bits of bytes $4, 5, 6, 7, 9, 10, 11, 13, 14, 15$. About $2^{78} \cdot 2^{78}/2^{80} = 2^{76}$ collisions are expected.
   (c) For each pair of ciphertexts that collide on the 80 bits (one encrypted under K_a and one under $K_{c'}$) do:
      i. Check that the ciphertext difference in bytes 2 and 3 may be caused by an input difference θ_0 to the S-box. If this is not the case, discard the pair. (about 1/4 of the pairs remain after this step).
      ii. Let $C_a \in S_l^a$ and $C_{c'} \in S_m^{c'}$ be the colliding ciphertexts, and P_a and $P_{c'}$ the respective plaintexts. Try all 2^{32} values for the bytes $0, 7, 10, 13$ of K^{11}, and partially decrypt the pair. If the difference of the partially decrypted values is in MB, access the list of pairs that corresponds to this subkey guess and the structures $S_l^b, S_m^{d'}$ from Step 2(a) (iii). Consider the pair P_a, $P_{c'}$, and each of the 2^{37} pairs of plaintexts under that subkey (as part of K_b and $K_{d'}$) as a candidate quartet (this leads to a total of 2^{44} candidate quartets with P_a and $P_{c'}$).
      iii. For any candidate quartet:
         – Check what is the key value for which the pairs $(C_a, C_{c'})$ and $(C_b, C_{d'})$ are partially decrypted to have a θ_0 difference in byte 8.

This operation can be performed efficiently using precomputed tables.
- Do the same for byte 12 of the ciphertext pairs.
- Check what is the key value for which the pairs (P_a, P_b) and $(P_{c'}, P_{d'})$ are partially encrypted to have a θ_0 difference in byte 2.
- Check what is the key value for which the pairs (P_a, P_b) and $(P_{c'}, P_{d'})$ are partially encrypted to have a $(x, 0, 0, 0)$ difference in bytes 1,5,11,12, where $x \to \theta_0$ by the S-box.
- Do the same for the bytes $2, 7, 8, 13$ of the plaintext pairs for the difference $(x \oplus \chi_0, \chi_1, \chi_2, \chi_3)$, where $x \to \theta_0$ by the S-box.

iv. If some quartet still remains at this point, assume that the subkey that it suggests is the correct one. Either perform an exhaustive key search on the remaining key bits, or use key ranking methods to find the right key.

4.6 Analysis of the Attack

We note that the first two tests of Step 2(c)(iii) can be done efficiently by computing for each pair independently the possible subkey values for which the condition is satisfied. Then, these tests are reduced to the problem of finding the intersection of these lists.

Once a right quartet is encountered, then it suggests the right value for 120 subkey bits (the relation between K_a and K_c suggests seven more bits of the key). Due to technical reasons, the time complexity of this search is 2^{112} (and not 2^{65} as might be expected).

For a given $K_{c'}$ value, out of the $2^{74} \cdot 2^{44} = 2^{118}$ quartets composed in Step 2(c)(ii), we expect 2^{60} quartets to reach Step 2(c)(iv) (a quartet has a probability of 2^{-7} to pass each of the first two tests, a probability of 2^{-8} to pass the third test, and a probability of 2^{-18} to pass each of the last two tests). The time complexity of the attack is $127 \cdot 2^{112} \cdot 2^{60} \approx 2^{179}$ encryptions.

If we take twice the data (i.e., 128 structures of 2^{72} encrypted under each key), we expect four right quartets. In that case, for any guess of the relation between K_a and $K_{c'}$, we shall perform the exhaustive search on the remaining key bits only if the same 120-bit value is suggested by two (or more) quartets. The time complexity in this case is dominated by the filtering done in Step 2(c), and is equal to 2^{118} encryptions for a given $K_{c'}$. Repeating the attack for every $K_{c'}$ (until one succeeds) has a worst-case time complexity of 2^{125} encryptions.

We conclude that the data complexity of our attack is 2^{87} chosen plaintexts, encrypted under 256 related keys (each key is used to encrypt 2^{79} values). The time complexity of the attack is 2^{125} 9-round encryptions.

The application of the attack to 10-round AES-256 is very similar. The attack AES-192). The data complexity of the attack is $2^{114.9}$ related-key chosen plaintexts (encrypted under 256 related keys) and the time complexity is $2^{171.8}$ encryptions.

5 Related-Key Boomerang Distinguisher for COCONUT98

COCONUT98 is a block cipher built according to the decorrelation methodology [22]. It contains two 4-round Feistel constructions with a decorrelation module in between. The differential (and linear) behavior of the decorrelation module is optimal in the sense that given a non-zero input difference, the probability to get any non-zero output difference is equal, when taken over all the keys. However, for any given key, the decorrelation module is an affine permutation.

The key schedule algorithm of COCONUT98 takes 256-bit keys, and divides them into two parts: a 128-bit subkey that enters the decorrelation module, and four 32-bit values, denoted by K_1, K_2, K_3, and K_4 that are used to derive the eight subkeys for the Feistel rounds. The subkeys of the first 4-round Feistel construction are: $K_1, K_1 \oplus K_3, K_1 \oplus K_3 \oplus K_4, K_1 \oplus K_4$, and the subkeys of the last 4-round Feistel construction are $K_2, K_2 \oplus K_3, K_2 \oplus K_3 \oplus K_4, K_2 \oplus K_4$.

The round function of the Feistel construction of COCONUT98 is

$$F_i((x,y)) = (y, x \oplus \phi((ROL_{11}(\phi(y \oplus k_i)) + c) \bmod 2^{32}))$$
$$\text{where} \qquad \phi(x) = (x + 256 \cdot S(x \& FF_x)) \bmod 2^{32}$$

where $S(\cdot)$ is an 8-bit to 24-bit S-box, c is a known 32-bit constant, & is the AND operator, and k_i is the 32-bit round subkey.

Our decomposition of COCONUT98 to sub-ciphers is as follows: the first sub-cipher consists of the first 4-round Feistel construction and the decorrelation module (denoted by DM). The second sub-cipher consists of the remaining 4-round Feistel construction.

For the first sub-cipher we use following related-key differential: Let the key difference be $\Delta K_0 = (0,0,z,z,0,0,0,0)$, then the differential is:

$$(z,0) \to (0,z) \to (z,0) \to (0,z) \to (0,z) \xrightarrow{DM} (z_1,z_2)$$

for some two unknown fixed (z_1, z_2), with probability 1. This is due to the fact that in each round where a difference z enters the round function, there is a subkey difference z to cancel it.

The related-key differential for the second sub-cipher is similar — the key difference is $\Delta K_1 = (0,0,z,z,0,0,0,0)$ and the second differential is:

$$(z,0) \to (0,z) \to (z,0) \to (0,z) \to (z,0)$$

with probability 1.

We note that $\Delta K_0 = \Delta K_1$. Hence, $K_c = K_a \oplus \Delta K_1 = K_a \oplus \Delta K_0 = K_b$, and $K_d = K_b \oplus \Delta K_1 = K_a$.

Thus, to find whether a given black box is COCONUT98, one can use the following algorithm:

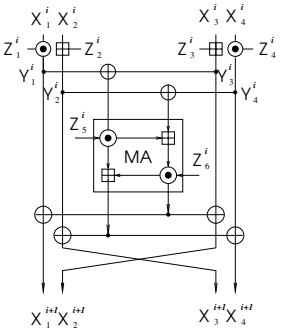

Fig. 4. One Round of IDEA

- Choose a non-zero value z of 32 bits. Choose a plaintext P_a.
- Ask the encryption of P_a under the key K_a, and denote the ciphertext by C_a. Ask the encryption of $P_b = P_a \oplus (z, 0)$ under the related-key K_b, and denote the ciphertext by C_b.
- Compute $C_c = C_a \oplus (z, 0)$ and $C_d = C_b \oplus (z, 0)$.
- Ask the decryption of C_c under $K_c = K_b$ to obtain P_c. Ask the decryption of C_d under $K_d = K_a$ to obtain P_d.
- If $P_c \oplus P_d = (z, 0)$, then output COCONUT98.

We note that if the key of the decorrelation module is such that the input difference of $(z, 0)$ to the decorrelation module remains $(z, 0)$ after the module, then we get a related-key differential with probability 1 for the entire COCONUT98 cipher. Otherwise, as $(z, 0) \not\to (z, 0)$ we get a related-key impossible differential (due to the miss in the middle attack [3, 2]).

We conclude that COCONUT98 can be easily distinguished using one related-key adaptive chosen plaintext and ciphertext quartet under two keys. By using two different z values, one for the first differential, and one for the second differential, the distinguisher remains the same in nature, but uses four keys instead of two.

6 Related-Key Rectangle Attack on IDEA

IDEA [21] is a 64-bit block cipher with 128-bit keys. It uses a composition of XOR operations, additions modulo 2^{16} and multiplications over $GF(2^{16} + 1)$. It has 8.5 rounds — 8 full rounds as described in Figure 4, and a final half-round consists of a layer of key additions and multiplications (Z_i^j are round subkeys). IDEA's key schedule is linear: each subkey is composed of bits of the key.

Since its introduction in 1991, IDEA has resisted a comprehensive cryptanalytic effort [10, 15, 2, 8, 13]. The best known attack against IDEA is on 5-round reduced version of IDEA. The attack uses 2^{24} chosen plaintexts and has a time complexity of 2^{116} encryptions [13]. IDEA also has several weak key classes. The

largest weak key class (identified by a boomerang technique) contains 2^{64} keys, and the membership test requires 2^{16} adaptive chosen plaintexts and ciphertexts and has a time complexity of 2^{16} encryptions [8].

Basically, our rectangle attack on 6.5-round IDEA uses a 6-round boomerang attack. The 6.5 rounds that we attack, start before the first MA function.

6.1 A Related-Key Boomerang Distinguisher for 5.5-Round IDEA

The 5.5-round distinguisher is used in rounds 2–6.5 (from the second round). The decomposition of the 5.5-round IDEA into sub-ciphers is as follows: The first sub-cipher has three rounds, while the second sub-cipher has two and a half rounds.

The input difference to round 2 (and the first sub-cipher) is $\alpha = (0_x, 0_x, 8000_x, 0_x)$. The key difference $\Delta K_0 = e_{25}$ (where e_i is a difference only in the i'th bit). The input difference is cancelled by the subkey difference, and the zero difference remains with probability 1 up to the MA of round 4. The key difference enters Z_5^4, leading to an unknown β difference after the MA. However, there are at most 2^{32} β values after the MA, and in the worst case all of them are equiprobable with probability 2^{-32}. As we use all these differentials simultaneously, we obtain that $\hat{p} = 2^{-16}$.

The second differential has a similar structure (but in the backward direction). The output difference is $\delta = (0_x, 0_x, 0100_x, 0_x)$ and the key difference is $\Delta K_1 = e_{75}$. In the decryption of a pair with difference δ, the key difference cancels the difference with probability 1/2. If this is the case, the zero difference remains through the decryption up till the first half round of round 5. This time, there are only 2^{16} possible γ values, and in the worst case, all of them are equiprobable [2] with probability 2^{-17}. Again, we use all of them simultaneously, and thus, $\hat{q}^2 = 2^{16} \cdot 2^{-34} = 2^{-18}$, and $\hat{q} = 2^{-9}$. We note that there are 8 more δ values (and respective ΔK_1 value) for which the attack can be mounted.

This leads to a distinguishing attack on a 5.5-round IDEA using $2^{51.6}$ quartets of adaptive chosen plaintexts and ciphertexts ($2^{49.6}$ values are to be encrypted/decrypted under four keys).

6.2 A Related-Key Boomerang Attack on 6-Round IDEA

Let K_a be the unknown key, $K_b = K_a \oplus e_{25}$, $K_c = K_a \oplus e_{75}$, and $K_d = K_a \oplus e_{25} \oplus e_{75}$. The boomerang attack on six rounds of IDEA (starting at the MA in round 1 till before the MA in round 7) is as follows:

1. For each guess of bits 64–95 of K_a set a counter initialized to 0.
2. Choose $2^{17.6}$ 32-bit value (r, t), and for each such value:
   - Choose a structure A of 2^{32} plaintexts of the form (x, y, z, w), such that $x \oplus z = t$ and $y \oplus w = r$.

[2] We have checked the claim experimentally. The values are not equiprobable, and the true value is $\hat{q} > 2^{-8.8}$. For more than 99% of the keys $\hat{q} > 2^{-8}$.

- Choose a structure B of 2^{32} plaintexts of the form $(x, y \oplus 8000_x, z, w)$, such that $x \oplus z = t$ and $y \oplus w = r$.
- Ask for the encryption of the structure A under K_a to receive A' and similarly ask for the encryption of B under K_b to receive B'.
- For any ciphertext in A' compute its XOR with the output difference of the second differential to obtain C'. Ask for the decryption of C' under K_c to obtain C.
- For any ciphertext in B' compute its XOR with the output difference of the second differential to obtain D'. Ask for the decryption of D' under K_d to obtain D.
- Insert all the plaintexts in C and D to a hash table indexed by the XOR value of the first and third word and by the XOR value of the second and fourth words.
- Examine a pair of colliding plaintexts (P_c, P_d). Let P_a be the plaintext that was encrypted, δ-shifted, and decrypted to P_c, and let P_b be the plaintext that was encrypted, δ-shifted and decrypted to P_d. For any guess of the bits 64–95 of K_a:
  (a) Partially encrypt P_a, P_b, P_c, P_d through the first MA. If the differences of the partial encryptions of P_a and P_b, and of the pair P_c and P_d are both α continue the analysis, if not so, try the next subkey.
  (b) Verify that the difference after a partial decryption of the respective ciphertext pairs is zero (bits 64–95 of the subkey contain the entire subkey which deals with the third word of the ciphertext). If this is the case, increment the counter of the subkey.

3. Output the subkey whose counter is maximal.

The structures are chosen so that in each pair of structures A, B there are 2^{32} pairs with input α after the first MA. For each such pair of structures we expect 2^{32} pairs of plaintexts (P_c, P_d) that are analyzed. Under random distribution assumptions, 2^{32} quartets from each pair of structures are encountered. However, most of them are discarded, and wrong quartet has probability of 2^{-32} to agree on the subkey of the first MA. Hence, we have about $2^{17.6}$ quartets in total, each suggesting 32-bit subkey value. The second filtering done, reduces this number by a factor of four.

We conclude that the attack suggests $2^{15.6}$ possible values to 32 bits of the key. As we expect four right quartets, then the right value is expected to be with the maximal counter with very high probability. The data complexity of the attack is $2^{51.6}$ adaptive chosen plaintexts and ciphertexts. The time complexity of the attack is $2^{51.6}$ MA evaluations which are equivalent to about 2^{48} encryptions.

6.3 A Related-Key Rectangle Attack on 6.5-round IDEA

The attack can be extended to a rectangle attack on 6.5-round IDEA. The 6.5 rounds starts after the first half round, and end after round 7. We use the same differentials as before. The algorithm of the attack is as follows:

- Choose $2^{25.8}$ values of the pair (r, t). Generate two structures of plaintexts for each value of (r, t) like in the boomerang attack.

- Ask the encryption of the structures under K_a and K_b, as before.
- Ask the encryption under K_c of each plaintext encrypted under K_a.
- Ask the encryption under K_d of each plaintext encrypted under K_b.
- For each guess of the subkey of the last MA, partially decrypt all ciphertexts under the guessed subkey, and call the boomerang attack on 6 rounds.

Out of the $2^{57.8}$ plaintexts encrypted under each key, we get about about $2^{51.6}$ pairs with the differences required for the previous attack. The attack has time complexity of $2^{32} \cdot (4 \cdot 2^{57.8}/13 + 2^{51.6}/13) = 2^{88.1}$ 6.5-round IDEA encryptions, and data complexity of $2^{59.8}$ chosen plaintexts under four related-keys.

7 Summary and Conclusions

In this paper we introduced related-key boomerang attacks and related-key rectangle attacks. The attacks use weaknesses in the key schedule algorithms of ciphers to achieve significant improvements over ordinary boomerang and rectangle distinguishers.

It is commonly believed that linearity of the key schedule is not a threat to the security of a block cipher if only its design (except for the key schedule) is moderate enough. Many strong block ciphers use linear or close to linear key schedule algorithms, e.g., AES, and IDEA. Despite the strong related-key requirements, our attacks show that it is important to maintain some non-linearity in the key schedule, even if the other components of the cipher seem strong enough.

References

1. Eli Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, Journal of Cryptology, vol. 7, number 4, pp. 229–246, Springer-Verlag, 1994.
2. Eli Biham, Alex Biryukov, Adi Shamir, *Miss in the Middle Attacks on IDEA and Khufu*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 124–138, Springer-Verlag, 1999.
3. Eli Biham, Alex Biryukov, Adi Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds*, Advances in Cryptology, proceedings of EUROCRYPT '99, Lecture Notes in Computer Science 1592, pp. 12–23, Springer-Verlag, 1999.
4. Eli Biham, Orr Dunkelman, Nathan Keller, *The Rectangle Attack – Rectangling the Serpent*, Advances in Cryptology, proceeding of EUROCRYPT 2001, Lecture Notes in Computer Science 2045, pp. 340–357, Springer-Verlag, 2001.
5. Eli Biham, Orr Dunkelman, Nathan Keller, *New Results on Boomerang and Rectangle Attacks*, proceeding of Fast Software Encryption 9, Lecture Notes in Computer Science 2365, pp. 1–16, Springer-Verlag, 2002.
6. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
7. Alex Biryukov, *The Boomerang Attack on 5 and 6-round AES*, preproceedings of Advanced Encryption Standard 4, available on-line at *http://www.esat.kuleuven.ac.be/∼abiryuko/*.
8. Alex Biryukov, Jorge Nakahara Jr., Bart Preneel, Joos Vandewalle *New Weak-Key Classes of IDEA*, proceedings of Information and Communications Security 4, Lecture Notes in Computer Science 2513, pp. 315–326, Springer-Verlag, 2002.

9. Alex Biryukov, David Wagner, *Slide Attacks*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 245–259, Springer-Verlag, 1999.
10. Johan Borst, Lars R. Knudsen, Vincent Rijmen, *Two Attacks on Reduced Round IDEA*, Advances in Cryptology, proceedings of EUROCRYPT '97, Lecture Notes in Computer Science 1233, pp. 1–13, Springer-Verlag, 1997.
11. Joan Daemen, Lars R. Knudsen, Vincent Rijmen, *The Block Cipher Square*, proceedings of Fast Software Encryption 4, Lecture Notes in Computer Science 1267, pp. 149–165, Springer-Verlag, 1997.
12. Joan Daemen, Vincent Rijmen *The design of Rijndael: AES — the Advanced Encryption Standard*, Springer-Verlag, 2002.
13. Hüseyin Demirci, Ali A. Selçuk, Erkan Türe, *A New Meet-in-the-Middle Attack on the IDEA Block Cipher*, proceedings of Selected Areas in Cryptography 2003, Lecture Notes in Computer Science 3006, pp. 117–129, Springer-Verlag, 2004.
14. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, Doug Whiting, *Improved Cryptanalysis of Rijndael*, proceeding of Fast Software Encryption 8, Lecture Notes in Computer Science 1978, pp. 213–230, Springer-Verlag, 2001.
15. Philip Hawkes, *Differential-Linear Weak Keys Classes of IDEA*, Advances in Cryptology, proceedings of EUROCRYPT '98, Lecture Notes in Computer Science 1403, pp. 112–126, Springer-Verlag, 1998.
16. Seokhie Hong, Jongsung Kim, Guil Kim, Sangjin Lee, Bart Preneel, *Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192*, proceedings of Fast Software Encryption 12, to appear.
17. Goce Jakimoski, Yvo Desmedt, *Related-Key Differential Cryptanalysis of 192-bit Key AES Variants*, proceedings of Selected Areas in Cryptography 2003, Lecture Notes in Computer Science 3006, pp. 208–221, Springer-Verlag, 2004.
18. John Kelsey, Tadayoshi Kohno, Bruce Schneier, *Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent*, proceedings of Fast Software Encryption 7, Lecture Notes in Computer Science 1978, pp. 75–93, Springer-Verlag, 2000.
19. John Kelsey, Bruce Schneier, David Wagner, *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA*, proceedings of Information and Communication Security 1997, Lecture Notes in Computer Science 1334, pp. 233–246, Springer-Verlag, 1997.
20. Jongsung Kim, Guil Kim, Seokhie Hong, Dowon Hong, *The Related-Key Rectangle Attack — Application to SHACAL-1*, proceedings of ACISP 2004, Lecture Notes in Computer Science 3108, pp. 123–136, Springer-Verlag, 2004.
21. Xuejia Lai, James L. Massey, *A Proposal for a New Block Cipher Encryption Standard*, Advances in Cryptology, proceeding of EUROCRYPT '90, Lecture Notes in Computer Science 473, pp. 389–404, Springer-Verlag, 1991.
22. Serge Vaudenay, *Provable Security for Block Ciphers by Decorrelation*, proceedings of Annual Symposium on Theoretical Aspects of Computer Science '98, Lecture Notes in Computer Science 1373, pp. 249–275, Springer-Verlag, 1998.
23. David Wagner, *The Boomerang Attack*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 156–170, Springer-Verlag, 1999.

On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions

John Black[1], Martin Cochran[1], and Thomas Shrimpton[2]

[1] Dept. of Computer Science, University of Colorado, Boulder CO 80309, USA
jrblack@cs.colorado.edu, Martin.Cochran@colorado.edu
www.cs.colorado.edu/~jrblack, ucsu.colorado.edu/~cochranm
[2] Dept. of Computer Science, Portland State University, Portland OR, 97207, USA
teshrim@cs.pdx.edu
www.cs.pdx.edu/~teshrim

Abstract. Fix a small, non-empty set of blockcipher keys $\mathcal{K}$. We say a blockcipher-based hash function is *highly-efficient* if it makes exactly one blockcipher call for each message block hashed, and all blockcipher calls use a key from $\mathcal{K}$. Although a few highly-efficient constructions have been proposed, no one has been able to prove their security. In this paper we prove, in the ideal-cipher model, that it is *impossible* to construct a highly-efficient iterated blockcipher-based hash function that is provably secure. Our result implies, in particular, that the Tweakable Chain Hash (TCH) construction suggested by Liskov, Rivest, and Wagner [7] is *not* correct under an instantiation suggested for this construction, nor can TCH be correctly instantiated by any other efficient means.

Keywords: Collision-resistant hash functions, tweakable blockciphers, provable security.

1 Introduction

Background. Essentially all modern hash functions are built by iterating a compression function according to the Merkle-Damgård paradigm [4, 10]. Moreover, these compression functions are almost always built from a blockcipher. Constructions like the Matyas-Meyer-Oseas (MMO) compression function [8] are explicit about their use of a blockcipher, but even so-called "dedicated" hashing primitives like MD5 and SHA-1 are in fact blockcipher-based. The SHA-1 compression function, for example, uses a 160-bit blockcipher that takes a 512-bit key; this blockcipher has been named SHACAL-1 [6].

This idea of building hash functions from blockciphers goes back more than 25 years. The earliest construction by Rabin [12] proposed to hash a message $M = m_1 m_2 \cdots m_\ell$ by fixing an initial value h_0 and computing $H(M) = \mathrm{DES}_{m_\ell}(\mathrm{DES}_{m_{\ell-1}}(\cdots(\mathrm{DES}_{m_1}(h_0))))$; effectively, this is a Merkle-Damgård construction with blockcipher-based compression function $f(h_{i-1}, m_i) = E_{m_i}(h_{i-1})$. Other constructions like Davies-Meyers [9] and MMO followed, but it was Preneel, Govaerts, and Vandewalle [11] who conducted the first systematic study

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 526–541, 2005.

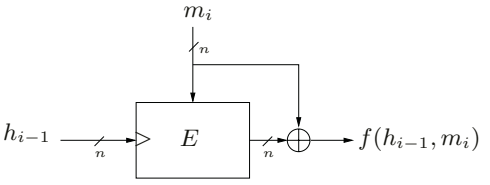

Fig. 1. The Matyas-Meyer-Oseas (MMO) compression function [8]. $E\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher; the hatch mark denotes the location of the key. Iterating this compression function results in a provably-secure hash function [2], however notice that the above compression function will be rekeyed each round

of blockcipher-based hash functions. They considered the security of 64 iterated constructions with compression functions of the form $f(h_{i-1}, m_i) = E_a(b) \oplus c$ where $a, b, c \in \{h_{i-1},\ m_i,\ h_{i-1} \oplus m_i,\ v\}$ for some fixed constant v. The analysis of PGV was attack-based, and schemes not broken by their attacks were deemed secure. Subsequently, Black, Rogaway and Shrimpton [2] considered these same 64 constructions using a proof-based approach. They showed that, in the ideal-cipher model, 20 of the 64 schemes are collision-resistant up to the birthday bound; Figure 1 gives one example.

Although provably secure, these 20 schemes could be viewed as inefficient in the following sense: in each, the blockcipher key is changed every round. For all conventional blockciphers, changing the key each round is undesirable since scheduling a new key entails a significant computational cost. It is natural to ask then if it is possible to achieve provable security without incurring this cost, and this question is the focus of our work.

Main Result. Fix a small, non-empty set of blockcipher keys $\mathcal{K}$. We term a blockcipher-based hash function *highly-efficient* if its compression function uses exactly one call to a blockcipher (ie, it is rate-1), and if the blockcipher uses only keys from $\mathcal{K}$. Since we can preschedule each key in $\mathcal{K}$, we enjoy a significant performance gain: key scheduling reduces to looking up a precomputed permutation. It is possible that those researchers who have worked on blockcipher-based hash functions over the past 25 years have considered highly-efficient constructions, but found attacks that broke these constructions, or were unable to prove their security. Indeed, the present authors also spent some time trying to find highly-efficient constructions without success. We now explain why.

One would like to construct a highly-efficient hash function that is provably collision resistant. If such a construction did exist, its underlying compression function could be constructed as follows (see Figure 2): let $f_1\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and $f_2\colon \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be arbitrary functions. We define $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ as

$$f(h_{i-1}, m_i) = f_2(h_{i-1}, m_i, E_K(f_1(h_{i-1}, m_i))),$$

where E_K is an n-bit blockcipher with key $K \in \mathcal{K}$ that is the output of a deterministic key-selection function g.

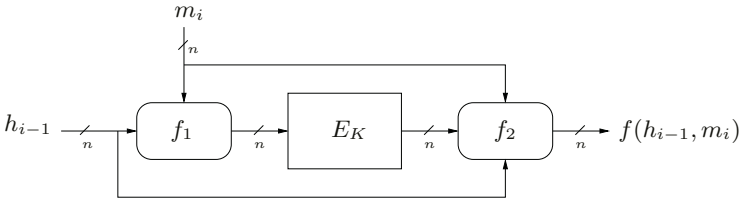

Fig. 2. The general compression function built from a blockcipher keyed from $\mathcal{K}$. Functions $f_1 \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and $f_2 \colon \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ are arbitrary; E_K is some n-bit blockcipher with key $K \in \mathcal{K}$ selected by a deterministic key-selection function g

It isn't hard see that this construction captures all possible rate-1 compression functions built from a blockcipher used in the forward direction and keyed from $\mathcal{K}$: both f_1 and f_2 may process every bit of the input to f in any arbitrary way. Notice that it isn't necessary to feed forward the output of f_1 to f_2, since f_2 can compute $f_1(h_{i-1}, m_i)$ itself. The key-selection function g must be deterministic and well-defined, but beyond this may depend on other message blocks, chaining values, the round number, or other parameters, provided it always returns a value from $\mathcal{K}$. These notions are made precise in Section 2. (Note that this approach does not capture *all* blockcipher-based hash functions with a fixed key-set, only those that are iterated via Merkle-Damgård and that use the blockcipher in the forward direction.)

In this paper we prove that any compression function constructed as just described *cannot* produce a provably collision-resistant hash function when iterated. Specifically, we show—in the ideal-cipher model—that for any functions f_1, f_2, there exists an information-theoretic adversary that finds a collision in the iterated function H^f in at most $|\mathcal{K}|(n + \lceil \lg(n) \rceil)$ blockcipher invocations. In fact, for many natural functions f_1, f_2 (like XOR) we find collisions in just $2|\mathcal{K}|$ blockcipher invocations. This is in stark contrast to the $\Theta(2^{n/2})$ expected invocations needed to produce a collision in the 20 rekeying constructions proven secure in [2]. Our impossibility proof uses a greedy algorithm that builds large numbers of messages along with their associated hash outputs. We prove that this algorithm builds a tree with height at most $n + \lceil \lg(n) \rceil$ containing at least $2^n(n + \lceil \lg(n) \rceil) + 1$ hash values, thereby yielding a collision on some level of the tree. We stress that our results should *not* be interpreted as giving practical attacks on all highly-efficient hash functions: our attacks have exponential running time. Instead we are exhibiting a proof that no highly-efficient iterated blockcipher-based hash functions can exist, in the model we have described.

Security of Tweak Chain Hash. Tweakable blockciphers were introduced by Liskov, Rivest, and Wagner [7]. They define a tweakable blockcipher as a map $\widetilde{E} \colon \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ where the inputs are called the *key*, the *tweak* and the *message*. We sometimes write $\widetilde{E}_K(T, M)$ instead of $\widetilde{E}(K, T, M)$. For any fixed $K \in \{0,1\}^k$ and $T \in \{0,1\}^t$, we require that $\widetilde{E}(K, T, \cdot)$ is a

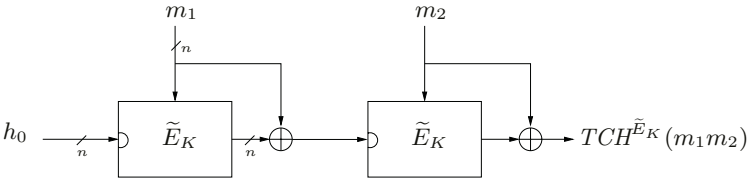

Fig. 3. Two rounds of the Tweak Chain Hash compression function. $\widetilde{E}_K$ is the tweakable blockcipher, K is a fixed public key; the arc denotes the location of the tweak

permutation on n bits. The idea is for the tweakable blockcipher to act like a normal blockcipher but with an extra (public) input, the tweak, which adds variability. The key may be expensive to schedule and to change, but changes to the tweak should be inexpensive. Security is defined as indistinguishability of a family of random permutations from $\widetilde{E}_K(\cdot,\cdot)$ with random key K, where the adversary controls the tweak and the message. See Section 2 for a formal definition.

Along with several other constructions, Liskov, Rivest, and Wagner suggest a new iterated hash-function construction built on tweakable blockciphers called the "Tweak Chain Hash" (TCH). This is a straightforward adaptation of the MMO construction into the tweakable setting: let $\widetilde{E}$ be a tweakable blockcipher with $t = n$, and fix a key $K \in \{0,1\}^k$. For any $M \in (\{0,1\}^n)^+$ write $M = m_1 \cdots m_\ell$ where each $|m_i| = n$, define $TCH^{\widetilde{E}_K}(M)$ as

function $TCH^{\widetilde{E}_K}(m_1 \cdots m_\ell)$
  for $i \leftarrow 1$ **to** ℓ **do** $h_i \leftarrow \widetilde{E}_K(h_{i-1}, m_i) \oplus m_i$
  return h_ℓ

where h_0 is a fixed constant, say 0^n. See Figure 3. A main motivation for TCH is efficiency: in each round the (expensive to change) key K remains fixed while the (cheap to change) tweak and message vary. One might therefore expect TCH to be substantially faster than MMO.

However the security of TCH is left as an open question. In the same paper, the authors propose two ways to create tweakable blockciphers from conventional blockciphers: one construction based on the CBC-MAC and one using universal hash families. So it is natural to wonder whether TCH is secure when built using either of these constructions. But inserting the second construction into TCH yields a fixed-key rate-1 hash function constructed from a *conventional* blockcipher and given our results above, we would expect that any such construction should be insecure. In fact we show something even stronger.

We demonstrate that using *either* tweakable-blockcipher construction from Liskov et al., the resulting TCH construction admits a simple attack. These attacks produce an infinite number of same-length colliding message pairs under TCH, regardless of the parameters chosen for the underlying tweakable blockcipher and regardless of the security model. Appealing to our main result, we further show in the ideal-cipher model that *any* tweakable blockcipher—built using one call to a conventional blockcipher—will yield an insecure TCH construction.

Our result does not, however, rule out TCH being secure when constructed from a tweakable blockcipher *primitive*, such as the Hasty Pudding cipher [13]. This is discussed further in Section 4.

Security Model. The standard-model assumption for blockciphers is that they are good pseudo-random permutations (PRPs) [9]. However this assumption is insufficient for proving the security of hash functions based on blockciphers; indeed, Simon has shown [15] that the PRP assumption alone is insufficient for secure blockcipher-based hashing. For this reason, all proofs of security for blockcipher-based hash functions have been done in the ideal-cipher model [2, 9, 10, 15, 16]. This model, which dates back to Shannon [14], treats a blockcipher as a random and independent permutation for each key. Some believe that modeling blockciphers in this way is not realistic: often we find correlations and biases in real blockciphers that one would not expect to see if the object were drawn uniformly from the family of all blockciphers with the same block and key size. Nonetheless, proofs of security in this model do have meaning: security is guaranteed against adversaries that ignore the structure of the underlying blockcipher.

Except for the two simple attacks given for TCH, all attacks in this paper are in the ideal-cipher model. We do not make any probabilistic assumptions nor do we depend on the permutivity of the blockciphers.

The normal measure of security for blockcipher-based hash functions is that they are secure against information-theoretic adversaries in the ideal-cipher model [2]. Our adversaries are therefore information-theoretic. While this may seem to be giving too much power to an adversary when thinking of real-world attacks, we once again stress that our goal is to demonstrate the nonexistence of *provably* collision-resistant schemes, not to give practically instantiable attacks. Indeed, it is clearly impossible to exhibit a practical attack given the generality of our setting: f_1 could itself be a collision-resistant hash function and a practical attack on the resulting hash function would imply a practical attack on f_1.

Message Lengths. Our definition for collision resistance will count as valid *any* pair of messages that produce the same hash value. Finding collisions in practice is often much harder than this due to techniques such as Merkle-Damgård strengthening [9]. In view of this, all attacks in this paper produce colliding messages of the same length, and therefore still apply even in the presence of such techniques.

2 Security Definitions

Basic Notions. Let k and n be positive integers. A *blockcipher* is a function $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ where for each $K \in \{0,1\}^k$ we require that $E_K(\cdot) = E(K,\cdot)$ is a permutation on $\{0,1\}^n$. Let $\mathrm{Perm}(n)$ be the set of all permutations on $\{0,1\}^n$, and let $\mathrm{Bloc}(k,n)$ be the set of all blockciphers $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. A function $f\colon \mathrm{Perm}(n) \times \mathcal{D} \to \{0,1\}^c$ is a (permutation-based) *compression function* if $\mathcal{D} = \{0,1\}^a \times \{0,1\}^n$ for some

$a \geq 1$ where $a + n \geq c$. If the program for computing f uses a single query $\pi(x)$ to compute $f^{\pi}(h, m) = f(\pi, h, m)$ then f is *rate-1*.

For non-empty sets $\mathcal{K} \subset \{0,1\}^k$ and $\mathcal{S} \subseteq \{0,1\}^*$, fix a deterministic *key-selection function* $g\colon \mathcal{S} \times \{0,1\}^n \times \{0,1\}^n \to \mathcal{S} \times \mathcal{K}$. Fix a constant h_0 and let $\sigma_0 = \varepsilon$. We define a (highly-efficient, blockcipher-based) *hash function* by the following program:

function $H[g,\mathcal{K}]^E(m_1 \cdots m_\ell)$
  for $i \leftarrow 1$ **to** ℓ **do**
    $(\sigma_i, K) \leftarrow g(\sigma_{i-1}, h_{i-1}, m_i)$
    $\pi \leftarrow E_K$
    $h_i \leftarrow f^{\pi}(h_{i-1}, m_i)$
  return h_ℓ

where $f\colon \mathrm{Perm}(n) \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a rate-1 compression function. This program computes a map $H[g,\mathcal{K}]^E\colon \mathrm{Bloc}(k,n) \times (\{0,1\}^n)^+ \to \{0,1\}^n$, and we say that $H[g,\mathcal{K}]^E$ is rate-1 because f is. Sometimes we will call $H[g,\mathcal{K}]^E$ the *iterated hash* of f, for obvious reasons.

When it is understood from context, we will omit the superscript π to f, and E to H. We will also often omit explicit reference to g and $\mathcal{K}$, simply writing H for $H^E[g,\mathcal{K}]$.

We write $x \xleftarrow{\$} S$ for the experiment of choosing a random element from the finite set S and calling it x. An *adversary* is an algorithm with access to one or more oracles, which we write as superscripts.

Collision resistance. To quantify the collision resistance of a highly-efficient, blockcipher-based hash function, we model the blockcipher as a randomly chosen $E \in \mathrm{Bloc}(k,n)$. An adversary A is given oracles for $E(\cdot,\cdot)$ and its inverse $E^{-1}(\cdot,\cdot)$, and wants to find a *collision* for H—that is, $M, M' \in \mathcal{D}$ where $M \neq M'$ but $H(M) = H(M')$. We look at the number of queries that the adversary makes and compare this with the probability of finding a collision.

Definition 1. [Collision resistance of a hash function] Fix $k, n > 0$, and let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Let $\mathcal{K} \subset \{0,1\}^k$ and $\mathcal{S} \subseteq \{0,1\}^*$ be non-empty sets, and let $g\colon \mathcal{S} \times \{0,1\}^n \times \{0,1\}^n \to \mathcal{S} \times \mathcal{K}$ be a key-selection function. Let $H[g,\mathcal{K}]$ be a highly-efficient, blockcipher-based hash function, $H[g,\mathcal{K}]\colon \mathrm{Bloc}(k,n) \times \mathcal{D} \to \{0,1\}^n$, and let A be an adversary. Then the advantage of A in finding collisions in H is the real number

$$\mathbf{Adv}^{\mathrm{coll}}_H(A) = \Pr\Big[E \xleftarrow{\$} \mathrm{Bloc}(k,n);\ (M, M') \xleftarrow{\$} A^{E(\cdot,\cdot),E^{-1}(\cdot,\cdot)} : M \neq M' \wedge H^E(M) = H^E(M')\Big]$$

For $q \geq 1$ we write $\mathbf{Adv}^{\mathrm{coll}}_H(q) = \max_A\{\mathbf{Adv}^{\mathrm{coll}}_H(A)\}$ where the maximum is taken over all adversaries that ask at most q oracle queries.

TWEAKABLE BLOCKCIPHERS. Fix $k, t, n > 0$. A *tweakable blockcipher* is a function $\widetilde{E}\colon \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ such that for any $K \in \{0,1\}^k$ and any $T \in \{0,1\}^t$ we are guaranteed that $\widetilde{E}(K,T,\cdot) = \widetilde{E}_K(T,\cdot)$ is a permutation on $\{0,1\}^n$. If we write $\widetilde{\pi} \xleftarrow{\$} \{0,1\}^t \times \mathrm{Perm}(n)$ we are choosing 2^t random permutations on $\{0,1\}^n$, one for each $T \in \{0,1\}^t$. The permutation associated to T is $\widetilde{\pi}(T,\cdot)$.

Definition 2. [Security of Conventional and Tweakable Blockciphers] Let $\widetilde{E}\colon \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher, and let A be an adversary. Then

$$\mathbf{Adv}^{\mathrm{prp}}_{E}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}\colon\ A^{E_K(\cdot)} = 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n)\colon\ A^{\pi(\cdot)} = 1]$$

$$\mathbf{Adv}^{\mathrm{tprp}}_{\widetilde{E}}(A) = \Pr[K \xleftarrow{\$} \{0,1\}^k\colon\ A^{\widetilde{E}_K(\cdot,\cdot)} = 1] - \Pr[\widetilde{\pi} \xleftarrow{\$} \{0,1\}^t \times \mathrm{Perm}(n)\colon\ A^{\widetilde{\pi}(\cdot,\cdot)} = 1]$$

Write $\mathbf{Adv}^{\mathrm{prp}}_{E}(q) = \max_A\{\mathbf{Adv}^{\mathrm{prp}}_{E}(A)\}$ and $\mathbf{Adv}^{\mathrm{tprp}}_{\widetilde{E}}(q) = \max_A\{\mathbf{Adv}^{\mathrm{tprp}}_{\widetilde{E}}(A)\}$ for $q \geq 1$ and where the maxima are taken over all adversaries that ask at most q oracle queries.

3 Hash Function Constructions and Attacks

We begin this section with a more detailed discussion of the generalized rate-1 blockcipher-based compression function shown in Figure 2, and of certain assumptions we might make in practice (though our later proofs will make no such assumptions). Next we consider attacks on the iterated hash of this compression function. The first attack is particularly efficient (it requires only $2|\mathcal{K}|$ blockcipher invocations) and we argue that it probably applies to many "reasonable" constructions for the compression function. The second is more general: it shows there cannot exist an iterated hash function based on this type of compression function that is provably collision resistant in our model.

GENERALIZED RATE-1 BLOCKCIPHER-BASED COMPRESSION FUNCTION. We consider any compression function f that is built in the following way. Let $f_1\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and $f_2\colon \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be arbitrary functions. We define $f\colon \mathrm{Perm}(n) \times (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^n$ as $f^{\pi}(h,m) = f_2(h,m,\pi(f_1(h,m)))$. See Figure 2 with $\pi = E_K$. We will not formally argue that this construction covers all possible $2n$ to n bit functions that call π at most once; this would take us quite far afield. Instead we give the following informal justification.

The function f takes two n-bit inputs, h and m. We make both of these inputs available to the "preprocessing" function f_1 and to the "postprocessing" function f_2. Additionally, f_2 has access to the output of π. We do not feed the output of f_1 to f_2 since f_2 is capable of recomputing f_1 itself. Similarly, we do not feed the output of f_2 back to f_1 since any computation performed by f_2

only on inputs h and m could have been computed by f_1; if the output of f_2 depends also on π then it cannot be fed back into f_1 (and thus π) because we are requiring f be rate-1.

Although f_1 and f_2 are fully arbitrary, we imagine that in practice they will be simple and fast-to-compute functions. In PGV [11], for example, these functions are never more complex than XOR. It would make little sense to have f_1 be, say, SHA-1 since our overall construction is itself aiming to be a cryptographic hash function. Nonetheless, our results continue to hold even for such far-fetched constructions: since our adversary is information-theoretic, it is able to find all $2n$-bit inputs that yield some particular n-bit output for f_1 in constant time.

The Two-Fiber Attack. We begin by describing a simple collision-finding attack called the "two-fiber attack" which works on many natural highly-efficient constructions, including all of the fixed-key constructions from [11]. In this attack the function f_1 has a certain property, which we call the "two-fiber property." We now explain.

Let $f_1^{-1}(i)$ represent the set $\{(h,m)\colon f_1(h,m) = i\}$. This is commonly called the *fiber* of f_1 under i, or the i-fiber. We now define the notion of a well-balanced fiber or function.

Definition 3. [Well-Balanced Fibers] Fix integer $n > 0$, and let $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a function. Then the fiber $f^{-1}(i)$ is *well-balanced* if each $h \in \{0,1\}^n$ appears exactly once as a first coordinate of some ordered pair of $f^{-1}(i)$. If every fiber of f is well-balanced, then we say that f is well-balanced.

An example of a well-balanced function is $f_1(h,m) = h \oplus m$. In fact, it's not hard to see that if $f_1(h,\cdot)$ is a permutation on $\{0,1\}^n$ for each $h \in \{0,1\}^n$ then f_1 will be well-balanced.

For the purposes of the present attack, we require only that there exist distinct $i_1, i_2 \in \{0,1\}^n$ such that $f_1^{-1}(i_1)$ and $f_1^{-1}(i_2)$ are well-balanced. If f_1 has two such fibers, we say that f_1 has the *two-fiber property* and the resulting attack is called the *two-fiber attack*. Notice that this property can be determined by the adversary without requiring any E-queries.

The attack is given in the theorem below. The idea behind the attack is that by doing just $2|\mathcal{K}|$ queries to E on points i_1 and i_2, an adversary can produce arbitrarily many same-length messages along with their hash values.

Theorem 1. [Two-Fiber Attack] Fix $k, n > 0$ and let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Fix $\mathcal{K} \subset \{0,1\}^k$, $\mathcal{S} \subseteq \{0,1\}^*$, and let $g\colon \mathcal{S} \times \{0,1\}^n \times \{0,1\}^n \to \mathcal{S} \times \mathcal{K}$ be a key-selection function. Let the compression function $f\colon \mathrm{Perm}(n) \times (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^n$ be defined as usual by $f^\pi(h,m) = f_2(h,m,\pi(f_1(h,m)))$, where f_1 has the two-fiber property. Finally, let hash function $H[g,\mathcal{K}]\colon \mathrm{Bloc}(k,n) \times \mathcal{D} \to \{0,1\}^n$ be the iterated hash of f. Then $\mathbf{Adv}_H^{\mathrm{coll}}(2|\mathcal{K}|) = 1$.

Proof. Let $i_1, i_2 \in \{0,1\}^n$, $i_1 \neq i_2$, be chosen such that $f_1^{-1}(i_1)$ and $f_1^{-1}(i_2)$ are well-balanced fibers. We now define $A^{E,E^{-1}}$, a collision-finding adversary for H.

First A makes $2|\mathcal{K}|$ queries to its left oracle at (K, i_1) and (K, i_2) for each $K \in \mathcal{K}$. With the resulting values, A grows a rooted tree T. Tree T will be annotated with node-labels and edge-labels; the edge-labels will represent message blocks and the node-labels will contain the intermediate hash values obtained by traversing T from the root to that node. Edges are added by specifying an ordered pair (u, v) of node labels where u is already in T and v is a new node with label v. Thus each edge-addition always creates a leaf. Each time we add an edge to T we will also specify the label for that edge.

Let the root of T be labeled h_0. Since $h_0 \in \{0,1\}^n$ and $f_1^{-1}(i_1)$ and $f_1^{-1}(i_2)$ are well-balanced fibers, then there exist distinct m_1, m_2 such that $i_1 = f_1(h_0, m_1)$ and $i_2 = f_1(h_0, m_2)$. Therefore A can now compute $x_1 = f(h_0, m_1)$ and $x_2 = f(h_0, m_2)$ without any oracle queries since $E_K(i_1)$ and $E_K(i_2)$ have been precomputed for any K that was output by g. If $x_1 = x_2$, then A halts returning the collsion m_1, m_2. If not, A adds an edge (h_0, x_1) labeled m_1 and an edge (h_0, x_2) labeled m_2. Then A continues at the leaves of T, doubling their number using the same technique as above; no additional oracle queries are required. This process is continued by A until a collision occurs. Since there are only 2^n possible output values for f and because the number of leaves doubles at each step, we are guaranteed that A will find a collision among the leaves within $n+1$ iterations of this process. ∎

Note that the proof holds even if E_K is not a permutation; we require only that E_K be a map from n bits to n bits. Also notice that the colliding messages produced by A are the same length; this means that a length-encoding scheme like MD-strengthening does not help avert the attack.

There are several obvious extensions to the two-fiber attack: for example, had we not insisted that messages be of the same length, a single well-balanced fiber would have sufficed. Also, if f_1 did not have the two-fiber property, perhaps it had ℓ fibers in which every $h \in \{0,1\}^n$ occurred at least twice among the first coordinates in those ℓ fibers. This would admit an analogous attack using $\ell|\mathcal{K}|$ oracle queries. Rather than pursue these ideas further, we instead proceed to the generalized attack that shows that $|\mathcal{K}|(n + \lceil \lg(n) \rceil))$ oracle queries are sufficient to find distinct same-length messages that collide for *any* generalized compression function.

Main Result. The central result of this paper is to show that *no* rate-1 compression function using blockcipher keys from a small fixed set $\mathcal{K}$ can give rise to a provably collision-resistant hash function when iterated. We show this by using at most $|\mathcal{K}|(n + \lceil \lg(n) \rceil)$ oracle queries to produce an overwhelming number of hash outputs that correspond to distinct messages. More specifically, our attack implements an algorithm to grow a tree of messages where the number of nodes in the tree at least doubles with each level added to it. We then show that the tree will have height at most $n + \lceil \lg(n) \rceil$ but with more than $2^n(n + \lceil \lg(n) \rceil)$ nodes which means there must exist a collision at some level of the tree.

Although the theorems below hold for all $n > 0$, we restrict our statements to $n \geq 8$ since small values are of no interest and addressing them would introduce

special cases into the proofs. Due to space constraints, we present here a proof sketch. The complete proof can be found in the full version of the paper [1].

Theorem 2. [General Attack] Fix $k > 0, n \geq 8$ and let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Fix $\mathcal{K} \subset \{0,1\}^k$, $\mathcal{S} \subseteq \{0,1\}^*$, and let $g\colon \mathcal{S} \times \{0,1\}^n \times \{0,1\}^n \to \mathcal{S} \times \mathcal{K}$ be a key-selection function. Define function $f\colon \mathrm{Perm}(n) \times (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^n$ as usual by $f^\pi(h,m) = f_2(h,m,\pi(f_1(h,m)))$, with f_1 and f_2 arbitrary. Let $H[g,\mathcal{K}]\colon \mathrm{Bloc}(k,n) \times \mathcal{D} \to \{0,1\}^n$ be the iterated hash of f. Then $\mathbf{Adv}^{\mathrm{coll}}_H(|\mathcal{K}|(n + \lceil \lg(n) \rceil)) = 1$.

Proof. We sketch the main ideas. Our goal is to grow a (rooted, directed) tree T where each edge and each node has a label. A path from the root to any node represents a message and its hash value; each edge along a given path is labeled with a message block, and the message associated to a given path is the concatenation of the edge-labels along that path. Each node is labeled with an n-bit chaining value; for any given path, the message associated to that path results in the hash value indicated at the terminal node of the path. We initialize T with a single node: the root, labeled by h_0. Thus far this is the same as it was for the two-fiber attack above.

For this sketch, we fix $|\mathcal{K}| = 1$, so g is trivial and K is fixed for all oracle queries. It is straightforward to generalize for larger $\mathcal{K}$.

Now, let $N = 2^n$ and for all $i \in [0..N-1]$ define $\mathcal{R}_i = \{(h,m,s)\colon h,m \in \{0,1\}^n \wedge f_1(h,m) = i)\}$. Here $s \in \{0,1\}^n \cup \{?\}$, and initially we set $s =?$ for all triples in the $\mathcal{R}_i$ above. The ? is a symbol indicating that the third coordinate of the triple is unknown; our goal is to have the third coordinate set to $f(h,m)$, but computing this requires an E-oracle query of (K,i) by A and we will use our oracle queries sparingly.

Note that once A performs the E-oracle query (K,i) he is able to fill in all third coordinates for triples in $\mathcal{R}_i$. And the goal is to grow T by as much as possible using as few queries as possible. Therefore A uses a greedy strategy: when choosing which (K,i) query to make next, he chooses the one that allows the largest number of new nodes to be added to T. When A makes this optimal query (K,i) he fills in the third coordinates of all triples in $\mathcal{R}_i$ and then adds edges to T with the corresponding edge-labels and node-labels.

As A proceeds, he adds at most one level to T for each query performed. The crux of the proof is to show that the number of nodes in T at least doubles each time A makes a query and extends T. In the full proof, we show that (except for the first query) a tree T with t nodes receives at least $t+1$ new nodes when A makes a query according to the greedy strategy above. The idea is a counting argument over the triples for those $\mathcal{R}_i$ that still contain a ?-symbol in their third coordinates. A simple inductive argument then shows that after ℓ queries we will have at least $2^\ell + 2^{\ell-1} - 1$ nodes in T.

Now letting $m = \lceil \lg(n) \rceil$, we see that after $n+m$ queries we are guaranteed at least $nN + 2^{m-1}N - 1$ nodes in $n+m+1$ levels of T. Ignoring the root node, this is $nN + 2^{m-1}N - 2$ nodes in $n+m$ levels. Since $n \geq 8$ then $2^{m-1} \geq m+1$ and

$$nN + 2^{m-1}N - 2 \geq (n+m)N + N - 2 > (n+m)N.$$

Thus there are more than $(n + \lceil \lg(n) \rceil)N$ nodes on $n + \lceil \lg(n) \rceil$ levels of T yielding a collision on some level. The same-length messages M and M' that collide under hash function H are extracted from T by traversing T from the root to each colliding node and reading off the edge labels that form the message blocks of M and M'. ∎

Interpreting the Result. We have used the ideal-cipher model for the blockcipher and endowed the adversary with limitless computational abilities. In this setting we were able to find an attack far more efficient than we can for known-secure constructions like MMO. However, we must realize that this model is not realistic in two ways: (1) When we plug a real blockcipher in for E, say 256-bit Rijndael, and fix a key-selection algorithm g and key-set $\mathcal{K}$, we do not then have a random object. We have a fixed public object that can be attacked via directed cryptanalysis. (2) If we attempt to mount the attacks described here, we will be using real computers with real computational limitations. Building a tree with $\Omega(2^n)$ nodes is not feasible for typical values of n. Of course collisions will appear long before the tree reaches this size, under reasonable probabilistic assumptions, but even a tree containing $\Omega(2^{n/2})$ nodes is impractical to store when n is (say) 160 or 256.

So one might reasonably ask if the attacks just shown are really of any concern at all. Perhaps we can use 256-bit Rijndael, fix a single key 0^{256}, and find some fast and simple functions f_1 and f_2 that do not admit any "obvious" attacks on the resulting iterated hash function. This may very well produce a collision-resistant hash function in the same sense that SHA-1 or RIPEMD-160 is thought to be collision resistant: no one has yet found collisions. However, we are taking a step backwards in this way of thinking because we are once again relying on the lack of effective attacks to give evidence of security. In a sense, we would be designing yet another primitive when we already have several primitives without any known attacks (at the time of this writing) and a longer established presence. But one thing we *can* guarantee about such an object is this: it will never admit a proof of security in the established model.

4 The Tweak Chain Hash

Tweakable blockciphers [7] are a map $\widetilde{E}\colon \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ where the inputs are called the "key," the "tweak," and the "message," respectively. We sometimes write $\widetilde{E}_K(T, M)$ instead of $\widetilde{E}(K, T, M)$. For any fixed $K \in \{0,1\}^k$ and $T \in \{0,1\}^t$, we require that $\widetilde{E}(K, T, \cdot)$ is a permutation on n bits. The idea is for the tweakable blockcipher to act like a normal blockcipher but with an extra (public) input, the tweak, which adds variability. The key may be expensive to schedule and to change, but changes to the tweak should be cheap. Security for a tweakable blockcipher was defined in Section 2.

In their paper, Liskov et al. give (among other things) two proposals for constructing tweakable blockciphers from conventional blockciphers, along with several other constructions for using tweakable blockciphers. Their paper suggests a new hash-function construction built on tweakable blockciphers called

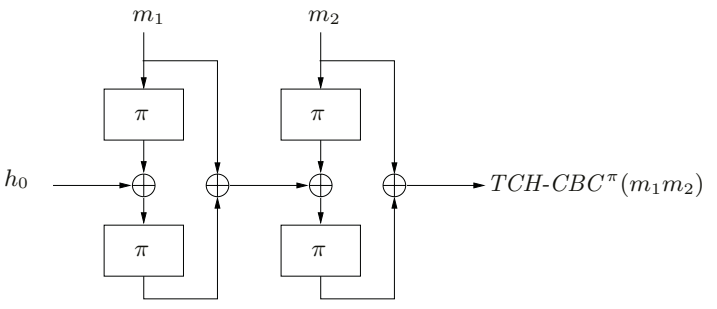

Fig. 4. Two rounds of the $TCH\text{-}CBC^{\pi}$ hash function. Function $\pi\colon \{0,1\}^n \to \{0,1\}^n$ is a fixed permutation. We can easily generate an infinite number of same-length message pairs that collide using this construction

the "Tweak Chain Hash" (TCH), defined as follows: for any $m \in (\{0,1\}^n)^+$ write $M = m_1 \cdots m_\ell$ where each $|m_i| = n$, define $TCH^{\widetilde{E}_K}(M)$ as

$$\begin{array}{l}\textbf{function } TCH^{\widetilde{E}_K}(m_1 \cdots m_\ell)\\ \quad \textbf{for } i \leftarrow 1 \textbf{ to } \ell \textbf{ do } h_i \leftarrow \widetilde{E}_K(h_{i-1}, m_i) \oplus m_i\\ \quad \textbf{return } h_\ell\end{array}$$

where h_0 is a fixed constant, say 0^n, and $\widetilde{E}$ is a tweakable blockcipher with $n = t$ and key K a constant; see Figure 3. Their is idea is that this construction should be faster than blockcipher-based constructions that rekey: the key K is fixed and only the tweak and message change for each message block digested. Since changing these two inputs should be cheap (ie, nothing equivalent to rescheduling a key should be required), each round of TCH should be faster than a round of, say, MMO. The authors leave the security of TCH as an open question. This is a question we aim to address in this section.

The First Attack: TCH-CBC. Liskov et al. give two provably-secure constructions of tweakable blockciphers from conventional blockciphers. The first construction is the CBC MAC of the two-block message $M \,\|\, T$. In other words, for a given blockcipher E they define $\widetilde{E}_K(T, M) = E_K(T \oplus E_K(M))$. They show that this construction is birthday-close to the underlying blockcipher E. That is, $\mathbf{Adv}^{\mathrm{tprp}}_{\widetilde{E}}(q) < \mathbf{Adv}^{\mathrm{prp}}_{E}(q) + q^2/2^n$. We call this the "CBC construction."

Taking the CBC construction and inserting it into the TCH construction seems like a natural try at building a collision-resistant hash function from a blockcipher. However, we immediately notice that the resulting TCH-CBC scheme is rate-1/2; that is, two blockcipher calls are required for each message block digested. (This means that our analysis from Section 3 does not apply because the compression function here is not rate-1.) This may in fact be more expensive than a rate-1 scheme that rekeys (like MMO). But TCH-CBC would be an interesting scheme nonetheless because it fixes the blockcipher key; no secure scheme has ever been exhibited that does this.

Unfortunately TCH-CBC is not collision resistant, as we now show. Fix a key K: this induces a fixed permutation that, for notational convenience, we name $\pi = E_K$. For any $M \in (\{0,1\}^n)^+$ write $M = m_1 \cdots m_\ell$ where each $|m_i| = n$, define $\mathit{TCH\text{-}CBC}^\pi(M)$ as

function $\mathit{TCH\text{-}CBC}^\pi(m_1 \cdots m_\ell)$
  for $i \leftarrow 1$ **to** ℓ **do** $h_i \leftarrow \pi(h_{i-1} \oplus\ \pi(m_i)) \oplus\ m_i$
  return h_ℓ

where as usual, h_0 is some fixed constant. See Figure 4. Now for a two-block message $M = m_1 m_2$ we have

$$\mathit{TCH\text{-}CBC}^\pi(M) = \pi(\pi(h_0 \oplus\ \pi(m_1)) \oplus\ m_1 \oplus\ \pi(m_2)) \oplus\ m_2.$$

Let $M_c^* = \pi^{-1}(c \oplus\ h_0) \,\|\, c$ and notice that $h(M_c^*) = h_0$ for any $c \in \{0,1\}^n$, yielding a large number of 2-block collisions. This idea can easily be generalized to generate collisions for messages of any even number of blocks > 2 as well.

The Second Attack: TCH-AXU. The second tweakable blockcipher construction proposed by Liskov et al. is based on the use of a universal hash family [3]. The flavor they used are known as ϵ-AXU$_2$ hash families. This is the preferred flavor because it leads to an efficient tweakable-blockcipher construction with good security. However, as we will see, plugging their construction into TCH allows a simple attack, and this attack does not depend on the ϵ-AXU$_2$ property.

Definition 4. [**ϵ-AXU$_2$ Hash Families**] Fix $n > 0$. We say a set of functions $\mathcal{U} = \{u\colon \{0,1\}^n \times \{0,1\}^n\}$ is ϵ-AXU$_2$ if for all $x, y, z \in \{0,1\}^n$ with $x \neq y$,

$$\Pr_{u \in \mathcal{U}}[u(x) \oplus u(y) = z] \leq \epsilon.$$

Now let E be a blockcipher, let $\mathcal{U}$ be an ϵ-AXU$_2$ hash family whose functions map n bits to n bits and define $\widetilde{E}_{K,u}(T, M) = E_K(M \oplus u(T)) \oplus u(T)$ where $K \in \{0,1\}^k$ and $u \in \mathcal{U}$. Liskov et al. show that $\mathbf{Adv}^{\mathrm{tprp}}_{\widetilde{E}}(q) < \mathbf{Adv}^{\mathrm{prp}}_{E}(q) + 3\epsilon q^2$. We call this the "AXU construction."

Let's try inserting the AXU construction into TCH and see if the resulting TCH-AXU construction is secure. Note that the AXU construction has a longer key since both the key for the underlying blockcipher and the function u must be specified. However, since TCH is a keyless object, we once again must fix both of these keys. Of course, fixing u means selecting some single function from $\mathcal{U}$, and since $\mathcal{U}$ is an ϵ-AXU$_2$ hash family, most of the functions in this set will be "good" in the sense that they will be injective or nearly injective. However, as we will see, the properties of the particular function u are irrelevant in our attack: it is effective no matter what n-bit to n-bit function is supplied.

Once we have selected a fixed key K and function u, we have a rate-1 fixed-key blockcipher-based hash function, and our results from Section 3 immediately tell us the construction is insecure. However, it is even worse than this: there is

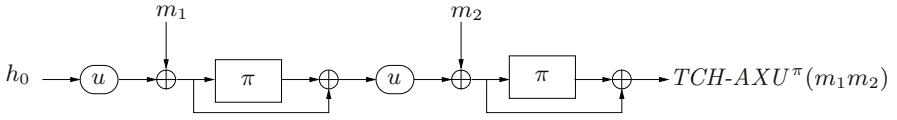

Fig. 5. Two rounds of the $TCH\text{-}AXU^{\pi}$ hash function. Function $\pi\colon \{0,1\}^n \to \{0,1\}^n$ is a fixed permutation and $u\colon \{0,1\}^n \to \{0,1\}^n$ is a fixed arbitrary function. We can easily generate an infinite number of same-length message pairs that collide using this construction

a very simple attack that yields an infinite number of same-length message pairs that collide, as we now demonstrate.

Fix a key K and a function u from the family $\mathcal{U}$. For notational convenience we name $\pi = E_K$. For any $M \in (\{0,1\}^n)^+$ write $M = m_1 \cdots m_\ell$ where each $|m_i| = n$, define $TCH\text{-}AXU^{\pi}(M)$ as

function $TCH\text{-}AXU^{\pi}(m_1 \cdots m_\ell)$
  for $i \leftarrow 1$ **to** ℓ **do** $h_i \leftarrow \pi(m_i \oplus u(h_{i-1})) \oplus m_i \oplus u(h_{i-1})$
  return h_ℓ

where as usual, h_0 is some fixed constant. See Figure 5. Now for a two-block message $M = m_1 m_2$ we have

$$TCH\text{-}AXU^{\pi}(M) = \pi(m_2 \oplus u(\pi(m_1 \oplus u(h_0)) \oplus m_1 \oplus u(h_0))) \\ \oplus m_2 \oplus u(\pi(m_1 \oplus u(h_0)) \oplus m_1 \oplus u(h_0)).$$

Let $M_c^* = u(h_0) \oplus c \parallel u(\pi(c) \oplus c)$ and notice that $h(M_c^*) = \pi(0)$ for any $c \in \{0,1\}^n$, yielding a large number of 2-block collisions. This idea can easily be generalized to generate collisions for messages of any even number of blocks > 2 as well.

The Security of TCH. The preceding two attacks do not imply that any tweakable blockcipher constructed as a mode on a conventional blockcipher will yield an easily-breakable TCH construction. It just so happened that the two modes given by the authors did fall to simple attacks. However, we can imagine other tweakable-blockcipher constructions where attacks on the resulting TCH are not so obvious. But the results of this paper tell us that if the tweakable blockcipher were constructed from a single call to a conventional blockcipher, the resulting TCH would not have a proof of security and would therefore have to be treated as a primitive.

A New Model. It is natural to ask whether TCH works under any model for tweakable blockciphers. And it's fairly clear that extending the ideal-cipher model to the tweakable setting does the trick: let $k, t, n \geq 1$ be numbers. Define $\mathrm{TBloc}(k,t,n)$ be the set of all tweakable blockciphers $\widetilde{E}\colon \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$. Choosing a random element of $\mathrm{TBloc}(k,t,n)$ means that for each $(K,T) \in \{0,1\}^k \times \{0,1\}^t$ one chooses a random permutation $E_K(T,\cdot)$.

For TCH, we require $t = n$ and we fix the key K to some constant. But this immediately reduces to MMO in the ideal-cipher model for conventional blockciphers, which was proven secure previously [2]. We have essentially lost the distinction between the key and the tweak since in our new ideal-tweakable-cipher model they are equivalent. The notion that the tweak is public and the key is secret has been lost. The notion that the tweak should be cheap to change while the key is normally expensive to change has similarly been lost.

What does provable security in the ideal-tweakable-cipher model mean? Notice that in each of the above attacks on TCH we exploited details of the construction of the underlying tweakable blockcipher. Had we treated these underlying objects as black boxes, we would have had no effective course of attack; we can therefore conclude that any attack on TCH must exploit the internal features of the tweakable blockcipher upon which it is constructed, meaning that perhaps a a tweakable blockcipher *primitive* might yield a secure TCH. The Hasty Pudding cipher is the only tweakable blockcipher primitive we know of [13]. Whether using Hasty Pudding in TCH yields an efficient collision resistant hash function is left as an open question, but we can be certain that any attacks on TCH-HP would require the cryptanalyst delve into the inner workings of the Hasty Pudding cipher.

5 Conclusion and Open Problems

Our results give strong evidence that we cannot build rate-1 collision-resistant hash functions from a blockcipher that uses only a small set of keys. Does this mean we are forced to accept constructions that change the key arbitrarily with each round if we want provable security? Not necessarily. Our results say nothing about schemes in this framework that rekey, say, every other round. It would be interesting to show sufficient conditions on how often the blockcipher must be rekeyed in order to maintain a good collision resistance bound. Alternatively, perhaps the key can be fixed in a non-Merkle-Damgård construction; the results of Gennaro et al. [5], although for a different security property than we considered here, may provide some insight. Or perhaps there is some relaxation of the model and weakening of the adversary that admit security proofs for highly-efficient blockcipher-based schemes. We leave these as open questions.

Acknowledgements

Thanks to Phillip Rogaway for suggesting to spell "blockcipher" as a single word (it saved typing a hyphen more than 20 times) and for various suggestions and comments on an early draft of this manuscript. Thanks as well to several Eurocrypt 2005 reviewers for their insightful suggestions. John Black's work was supported by NSF CAREER-0240000 and a gift from the Boettcher Foundation. Part of this work was conducted while Tom Shrimpton was at UC Davis and was supported by NSF 0208842, NSF 0085961, and a gift from Cisco Systems.

References

1. Black, J., Cochran, M., and Shrimpton, T. On the impossibility of highly efficient blockcipher-based hash functions. Full version of this paper, www.cs.colorado.edu/~jrblack, 2005.
2. Black, J., Rogaway, P., and Shrimpton, T. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *Advances in Cryptology – CRYPTO '02* (2002), vol. 2442 of *Lecture Notes in Computer Science*, Springer-Verlag.
3. Carter, L., and Wegman, M. Universal hash functions. *J. of Computer and System Sciences*, 18 (1979), 143–154.
4. Damgård, I. A design principle for hash functions. In *Advances in Cryptology – CRYPTO '89* (1990), G. Brassard, Ed., vol. 435 of *Lecture Notes in Computer Science*, Springer-Verlag.
5. Gennaro, R., Gertner, Y., Katz, J., and Trevisan, L. Bounds on the efficiency of generic cryptographic constructions, 2005. To appear in the *SIAM Journal on Computing*.
6. Handschuh, H., Knudsen, L., and Robshaw, M. Analysis of SHA-1 in encryption mode. In *Advances in Cryptology – CT-RSA '01* (2001), D. Naccache, Ed., Lecture Notes in Computer Science, Springer-Verlag, pp. 70–83.
7. Liskov, M., Rivest, R., and Wagner, D. Tweakable block ciphers. In *Advances in Cryptology – CRYPTO '02* (2002), M. Yung, Ed., Lecture Notes in Computer Science, Springer-Verlag, pp. 31–46.
8. Matyas, S., Meyer, C., and Oseas, J. Generating strong one-way functions with cryptographic algorithms. *IBM Technical Disclosure Bulletin 27*, 10a (1985), 5658–5659.
9. Menezes, A., van Oorschot, P., and Vanstone, S. *Handbook of Applied Cryptography*. CRC Press, 1996.
10. Merkle, R. One way hash functions and DES. In *Advances in Cryptology – CRYPTO '89* (1990), G. Brassard, Ed., vol. 435 of *Lecture Notes in Computer Science*, Springer-Verlag.
11. Preneel, B., Govaerts, R., and Vandewalle, J. Hash functions based on block ciphers: A synthetic approach. In *Advances in Cryptology – CRYPTO '93* (1994), Lecture Notes in Computer Science, Springer-Verlag, pp. 368–378.
12. Rabin, M. Digitalized signatures. In *Foundations of Secure Computation* (1978), R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, Eds., Academic Press, pp. 155–168.
13. Schroeppel, R., and Orman, H. The hasty pudding cipher. AES candidate submitted to NIST, 1998.
14. Shannon, C. Communication theory of secrecy systems. *Bell Systems Technical Journal 28*, 4 (1949), 656–715.
15. Simon, D. Finding collsions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT '98* (1998), Lecture Notes in Computer Science, Springer-Verlag, pp. 334–345.
16. Winternitz, R. A secure one-way hash function built from DES. In *Proceedings of the IEEE Symposium on Information Security and Privacy* (1984), IEEE Press, pp. 88–90.

Public Traceability in Traitor Tracing Schemes

Hervé Chabanne[1], Duong Hieu Phan[2], and David Pointcheval[2]

[1] SAGEM, Eragny, France
[2] CNRS/ENS, Computer Science Department, Paris, France
http://www.di.ens.fr/users/{phan,pointche}

Abstract. Traitor tracing schemes are of major importance for secure distribution of digital content. They indeed aim at protecting content providers from colluding users to build pirate decoders. If such a collusion happens, at least one member of the latter collusion will be detected. Several solutions have already been proposed in the literature, but the most important problem to solve remains having a very good ciphertext/plaintext rate. At Eurocrypt '02, Kiayias and Yung proposed the first scheme with such a constant rate, but still not optimal. In this paper, granted bilinear maps, we manage to improve it, and get an "almost" optimal scheme, since this rate is asymptotically 1. Furthermore, we introduce a new feature, the "public traceability", which means that the center can delegate the tracing capability to any "untrusted" person. This is not the first use of bilinear maps for traitor tracing applications, but among the previous proposals, only one has remained unbroken: we present an attack by producing an anonymous pirate decoder. We furthermore explain the flaw in their security analysis. For our scheme, we provide a complete proof, based on new computational assumptions, related to the bilinear Diffie-Hellman ones, in the standard model.

1 Introduction

The secure distribution of digital content to a set of subscribers is an important application of global networking (e.g. pay-per-view television.) There are two main types of schemes in the literature to deal with this topic: broadcast encryption schemes, which enable a center to prevent a set of users from recovering the broadcasted information; and traitor tracing schemes, which enable the center to trace users who collude to produce pirate decoders. Both types of schemes can be trivially combined by XOR'ing the results as shown in [7, 8]. There are also several works considering efficient combinations of the two attributes of *broadcast capability* and *traceability* [9, 10, 19, 17]. This paper focuses on the *traceability property*. As mentioned in the seminal paper on *traitor tracing* of Chor *et .al* [7, 8], a c-traitor tracing scheme should guarantee that:

1. either the cleartext information itself is continuously transmitted to the enemy by a traitor;
2. or any captured pirate decoder will correctly identify a traitor and will protect the innocent even if up to c traitors collude.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 542–558, 2005.

There is indeed no technical way to prevent a pirate from decoding and forwarding the stream to a user. But this would be rather expensive and commercially unattractive. Therefore, traitor tracing schemes deal with the traceability of pirate decoders only.

1.1 Transmission Rates

A direct solution to the traitor tracing problem is to give to each subscriber an individual key and encrypt the data separately under each key. But this is extremely inefficient because this means that the total size of the broadcast ciphertext is at least n times the size of the plaintext, where n is the number of authorized users: the ciphertext/plaintext rate is thus greater than n. The transmission rate [14] has a quite important practical impact. It actually collects three parameters: ciphertext rate, encryption-key rate and user-key rate, which are respectively the ratio of the size of ciphertext, encryption-key and user-key over the size of the plaintext (in an asymptotic way.) We thus have two main categories for traitor tracing schemes:

1. Schemes with *constant* transmission rate [14]. They are well-suited to encrypt large messages. Another interesting advantage of these schemes is the efficient black-box traceability. This means that the tracing procedure does not have to open the pirate decoder, but just to interact with it. On the other hand, the constant transmission rate is asymptotically achieved, and thus for large plaintexts only (this is due to the use of collision-secure codes.)
2. Schemes with *no constant* transmission rate [4, 2, 15]. The main advantage of these schemes is about their relatively small size of admissible plaintexts. However, the transmission rate is often linear w.r.t the maximal number of colluders. Furthermore, in these schemes, there is no efficient black-box traitor tracing. It is possible to do black-box traitor tracing [2], but it is shown that the algorithm is non-realistic because of the complexity which is larger than the binomial of n and c, where n is the number of users and c is the maximal number of colluders.

According to the context, one may use a scheme from the first category or a scheme from the second one: if one wants to distribute large messages, the first category is much more suitable, however if one simply wants to exchange a session key, which size is relatively small, the second category may be better from efficiency point of view, but the actual security can be discussed because of the inefficient black-box tracing procedure. In this paper, we further improve the transmission rate of the unique above constant transmission rate scheme [14].

1.2 Traceability

In all known traitor tracing schemes, only the center, owning some crucial private information, can execute the tracing procedure: delegation is not possible, unless the center discloses private information allowing to trace, but also to create new *anonymous* decoders, which is not reasonable.

However, such a delegation could be a quite interesting feature: if the center is the only server able to run the tracing procedure, a bottleneck may appear because of a possibly large number of pirate decoders.

This paper thus introduces a new property, called *public traceability*: the tracing procedure can be publicly done, by simply providing the tracing information, which just helps to trace, but nothing else.

1.3 Bilinear Maps

Let us now turn to the tool recently introduced in cryptographic protocols by Joux [13]: the use of some specific bilinear maps, such as the modified Weil pairing or the Tate pairing. They have already been widely used to achieve new features, such as identity-based cryptosystems [3], or to improve the efficiency of some schemes [1]. However, such particular properties could be used by adversaries too, in order to break underlying schemes such as the attacks from [20] on the traitor tracing scheme proposed in [16].

In this paper, we show these two sides of the use of bilinear maps. On the one hand, we show how the pairings can be used for improving a traitor tracing scheme, in two directions. It indeed helps to get a more efficient scheme as well as the new feature of *public traceability*. On the other hand, we show that the adversaries can also take advantage of them in some schemes: we present an attack against the only unbroken traitor tracing scheme based on pairings [20].

1.4 Contribution

At Eurocrypt '02, Kiayias and Yung [14] proposed a new traitor tracing scheme (named KY in the following) with constant transmission rates: the ciphertext rate is 3, the encryption-key rate is 4 and the user-key rate is 2.

In this paper we propose a scheme which further improves them: the ciphertext rate is reduced to 1 (asymptotically), which is optimal; the encryption-key rate is reduced to 1; and the user-key rate is kept unchanged. As already noticed, these transmission rates are considered in the multi-user setting, when the number of users is large, and when the size of the plaintext is large too. Above improvements are achieved, while still keeping the two extremely desirable properties, as in the KY scheme:

- *public-key* traitor tracing, where any third party is able to send secure messages to the set of subscribers;
- efficient *black-box* traitor tracing in which the tracing procedure can be accomplished without opening the pirate decoder.

We furthermore introduce a new quite interesting functionality: the *public traceability*. In all previous traitor tracing schemes, only the center, owning some crucial private information, could execute the tracing procedure. In our scheme, the center can publish some information in such a way that every one can do the tracing procedure, at least the interactive part with the pirate decoder.

As already said, pairings are of great help to achieve this goal. But care is required. To the best of our knowledge, only one such a scheme based on

pairings has remained unbroken: the scheme proposed by To, Safavi-Naini and Zhang [20] (named TSZ in the following). In this paper we show an attack on the TSZ scheme: we exhibit a way to produce an anonymous pirate decoder, while they provided a security proof. We thereafter explain where is the flaw in their tracing algorithm.

2 TSZ: The To, Safavi-Naini and Zhang's Scheme

Mitsunari, Sakai and Kasahara [16] proposed the first traitor tracing scheme using the bilinear maps. One year later, To, Safavi-Naini and Zhang [20] presented an attack and tried to repair it. Unfortunately, this modification is not correct either. Let us first review it, then we present an attack. This scheme and the attack will help to understand later our new construction which is a combination of the TSZ scheme and the KY scheme, taking advantage of the best of each.

2.1 Description of the Scheme

The TSZ scheme uses a bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$, where $\mathcal{G}_1, \mathcal{G}_2$ are groups of prime order q (see section 3 for a brief review.)

Initialization: two arbitrary random generators $P, Q \in \mathcal{G}_1$ and a unitary polynomial with coefficients in $\mathbb{Z}_q$ of degree $2k-1$:

$$f(x) = a_0 + a_1 x + \ldots + a_{2k-2} x^{2k-2} + x^{2k-1}.$$

Let $Q_0 = a_0 Q$, $Q_1 = a_1 Q$, $\ldots$, $Q_{2k-2} = a_{2k-2} Q$ and $g = \hat{e}(P, Q) \in \mathcal{G}_2$.

Private key of the center: the generator P, and the polynomial f.

Encryption key: the tuple $(g, Q, Q_0, Q_1, \ldots, Q_{2k-2})$.

User key (for user u): $K_u = f(u)^{-1} P$.

Encryption Algorithm: one generates a random $r \in \mathbb{Z}_q$, then the session key $s \in \mathcal{G}_2$ is encrypted into: $c = (sg^r, rQ, rQ_0, \ldots, rQ_{2k-2})$.

Decryption Algorithm: user u first computes g^r, granted K_u, and then recovers s. Indeed, $g^r = \hat{e}(K_u, rQ_0) \times \ldots \times \hat{e}(u^{2k-2} K_u, rQ_{2k-2}) \times \hat{e}(u^{2k-1} K_u, rQ)$.

2.2 Attack

In [20], authors showed that nobody can build an anonymous decoder, even a collusion of registered users. Here, we explain how a unique user can build such an anonymous decoder: user u chooses random elements $z_0, z_1, \ldots, z_{2k-2}$ in $\mathbb{Z}_q$ and produces the following decoder:

- $X_i = u^i K_u + z_i Q$, for i from 0 to $2k-2$.
- X_{2k-1} is determined by the relation:

$$X_{2k-1} = u^{2k-1} K_u - (z_0 Q_0 + z_1 Q_1 + \ldots + z_{2k-2} Q_{2k-2}). \tag{1}$$

User u can then publish $X_0, X_1, \ldots, X_{2k-1}$, which provides everyone with the ability of recovering $g^r = \hat{e}(X_0, rQ_0) \times \ldots \times \hat{e}(X_{2k-2}, rQ_{2k-2}) \times \hat{e}(X_{2k-1}, rQ)$.

2.3 Flaw in the Security Analysis

While authors provided a tracing procedure, we now show that our above decoder, which only uses $X_0, \ldots, X_{2k-2}, X_{2k-1}$, cannot trace back user u. First, $X_0, \ldots, X_{2k-2}, X_{2k-1}$ satisfy the following relation:

$$\sum_{0}^{2k-1} a_i X_i = \left(\sum_{0}^{2k-2} a_i u^i\right) K_u + \left(\sum_{0}^{2k-2} a_i z_i\right) Q + u^{2k-1} K_u - \left(\sum_{0}^{2k-2} z_i a_i\right) Q$$
$$= f(u) K_u = P. \quad (2)$$

Remark also that for each user, and all i ($i = 0, \ldots, 2k-2$), the application, from $\mathbb{Z}_q$ to $\mathcal{G}_1$, which maps z_i to X_i, is a bijection. Therefore, instead of choosing $z_0, z_1, \ldots, z_{2k-2}$, one can randomly choose $X_0, \ldots, X_{2k-2}$ in $\mathcal{G}_1$, which uniquely defines the tuple $(z_0, z_1, \ldots, z_{2k-2})$. Thereafter, X_{2k-1} is also uniquely determined by the relation (1). It also satisfies the relation (2). Hence, one can formally define it from the latter relation: it thus clearly does not depend on u.

As a consequence, one easily sees that all the users would produce the same set of pirate decoders, with parameters $(X_0, X_1, \ldots, X_{2k-2}, X_{2k-1})$, so that X_0, X_1, $\ldots, X_{2k-2}$ are randomly chosen in $\mathcal{G}_1^{2k-1}$, while X_{2k-1} is defined according to the relation (2).

Note that this attack is quite different from the one in [20]. Our pirate decoder indeed combines informations of the user-key, together with the *public information* of the system. The latter part points out the flaw in the tracing algorithm from [20], which works as follows: for a suspect set of users $A = \{u_1, \ldots, u_t\}$ (whose size is up to k), they construct another polynomial $f'(x) = f(x) + \alpha \times (x - u_1) \times \ldots \times (x - u_t)$. For any user in the set A, his key in the scheme using f (named $\mathsf{Scheme}(f)$) and the one in the scheme using f' (named $\mathsf{Scheme}(f')$) are identical. For this reason, they claimed that if the colluders are in the set A, then any pirate decoder produced by them in $\mathsf{Scheme}(f)$ is also a pirate decoder in $\mathsf{Scheme}(f')$. Accordingly, this decoder will decrypt a ciphertext in $\mathsf{Scheme}(f')$ as it would be in $\mathsf{Scheme}(f)$. Therefore, by sending a decryption query to the decoder, the center can easily detect whether the set of colluders is included in A or not.

Unfortunately, their argument is not correct. If the construction of the pirate decoder depends only on the user-keys of the colluders, their tracing algorithm works well. But if the construction depends on the public information too (which are of course available to the colluders), the tracing procedure fails, as shown above.

3 Bilinear Maps and Computational Assumptions

3.1 Bilinear Maps

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two groups of order q, for some large prime q. We use in our system a bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$, which must satisfy the following properties:

Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathcal{G}_1$ and all $a, b \in \mathbb{Z}_q$;
Non-degenerated: The map does not send all pairs in $\mathcal{G}_1 \times \mathcal{G}_1$ to the unit in $\mathcal{G}_2$;
Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any elements $P, Q \in \mathcal{G}_1$.

A bilinear map satisfying the three above properties is said to be an *admissible bilinear map*. Throughout the paper we view $\mathcal{G}_1$ as an additive group and $\mathcal{G}_2$ as a multiplicative group. Remark that since $\mathcal{G}_1, \mathcal{G}_2$ are groups of prime order and $\hat{e}$ is non-degenerated, if P is a generator of $\mathcal{G}_1$ then $\hat{e}(P, P)$ is a generator of $\mathcal{G}_2$.

Example: The modified Weil pairing or the Tate pairing can be used to construct an admissible bilinear map that satisfies the three above properties.

3.2 Computational Assumptions

In this section, we review some well-known problems such as the computational bilinear Diffie-Hellman problems. We also propose new problems, we believe to be hard to solve. Relations claimed in propositions are provided in the full version [6]. They will be used in the next sections, in the security analysis of our scheme.

Classical Assumptions and Variants. We first review the most classical problems in $\mathcal{G}_1$.

CDH – the computational Diffie-Hellman problem in $\mathcal{G}_1$:
Given (P, aP, bP) for some $a, b \in \mathbb{Z}_q^\star$, output abP.
CBDH^1 – the computational bilinear Diffie-Hellman problem in $\mathcal{G}_1$:
Given (P, aP, bP, cP) for some $a, b, c \in \mathbb{Z}_q^\star$, output $abcP$.
DBDH^1 – the decisional bilinear Diffie-Hellman problem in $\mathcal{G}_1$:
Given (P, aP, bP, cP, U) for some $a, b, c \in \mathbb{Z}_q^\star$ and $U \in \mathcal{G}_1$, output yes if $U = abcP$ and no otherwise.

We now introduce modified versions of the two above Bilinear Diffie-Hellman problems. They are actually particular cases, where $b = c$. We then provide some relations between them and the usual CDH problem.

CBDH^1-M – the modified computational bilinear Diffie-Hellman problem in $\mathcal{G}_1$:
Given (P, aP, bP) for some $a, b \in \mathbb{Z}_q^\star$, output ab^2P.
DBDH^1-M – the modified decisional bilinear Diffie-Hellman problem in $\mathcal{G}_1$:
Given(P, aP, bP, U) for some $a, b \in \mathbb{Z}_q^\star$ and $U \in \mathcal{G}_1$, output yes if $U = ab^2P$ and no otherwise.

Proposition 1. *The* CBDH^1-M *problem is at least as hard as the* CBDH^1 *problem, which is at least as hard as the usual* CDH *problem:*

$$(\mathsf{Succ}_{\mathcal{G}_1}^{\mathsf{CBDH}^1\text{-}\mathsf{M}}(t))^2 \leq \mathsf{Succ}_{\mathcal{G}_1}^{\mathsf{CBDH}^1}(t) \leq \mathsf{Succ}_{\mathcal{G}_1}^{\mathsf{CDH}}(t).$$

Pairing-Based Problems. We now review the bilinear Diffie-Hellman problems, all in $\mathcal{G}_1$ and $\mathcal{G}_2$, with the admissible map $\hat{e}$ (we thus omit them in the notation.)

CBDH^2 – the computational bilinear Diffie-Hellman problem:
Given (P, aP, bP, cP) for some $a, b, c \in \mathbb{Z}_q^\star$, output g^{abc}, where $g = \hat{e}(P, P)$.

DBDH^2 – the decisional bilinear Diffie-Hellman problem:
Given (P, aP, bP, cP, Z) for some $a, b, c \in \mathbb{Z}_q^\star$ and $U \in \mathcal{G}_2$, output yes if $Z = g^{abc}$ and no otherwise, where $g = \hat{e}(P, P)$.

CBDH^2-E – the extended computational bilinear Diffie-Hellman problem:
Given (P, aP, bP, cP, ab^2P) for some $a, b, c \in \mathbb{Z}_q^\star$, output g^{cb^2}, where $g = \hat{e}(P, P)$.

DBDH^2-E – the extended decisional bilinear Diffie-Hellman problem:
Given $(P, aP, bP, cP, ab^2P, Z)$ for some $a, b, c \in \mathbb{Z}_q^\star$ and $U \in \mathcal{G}_2$, output yes if $Z = g^{cb^2}$ and no otherwise, where $g = \hat{e}(P, P)$.

We furthermore introduce a slight variant of the CBDH^2, in order to get more confidence in the above CBDH^2-E problem:

CBDH^2-V – a variation of the computational bilinear Diffie-Hellman problem:
Given $(P, aP, bP, cP, a(a^2 - b^2)P, b(a^2 - b^2)P)$ for some $a, b, c \in \mathbb{Z}_q^\star$, output g^{abc}, where $g = \hat{e}(P, P)$.

Proposition 2. *The* CBDH^2-E *problem is at least as hard as the* CBDH^2-V *problem:*

$$(\mathsf{Succ}^{\mathsf{CBDH}^2\text{-}\mathsf{E}}_{\hat{e},\mathcal{G}_1,\mathcal{G}_2}(t))^2 \leq \mathsf{Succ}^{\mathsf{CBDH}^2\text{-}\mathsf{V}}_{\hat{e},\mathcal{G}_1,\mathcal{G}_2}(t).$$

Mixed Problems. Let us now introduce new problems which involve elements from $\mathcal{G}_1$ and $\mathcal{G}_2$, still with the admissible map $\hat{e}$ (we thus omit them in the notation.)

MCDH – the mixed computational Diffie-Hellman problem:
Given (P, aP, a^2P, g^b) for some $a, b \in \mathbb{Z}_q^\star$, where $g = \hat{e}(P, P)$, output g^{ba^2}.

MDDH – the mixed decisional Diffie-Hellman problem:
Given (P, aP, a^2P, g^b, Z) for some $a, b \in \mathbb{Z}_q^\star$ and $Z \in \mathcal{G}_2$, where $g = \hat{e}(P, P)$, output yes if $Z = g^{ba^2}$ and no otherwise.

4 The Basic Building Block: The Two-User Case

4.1 The Assumptions for Our Scheme

We have introduced several new problems, which will simplify the security analysis of our proposal. Let us sum up which assumptions will be really needed, according to the security level.

Traitor Tracing. Let us first consider the semantic security of the encryption scheme. In the random-oracle model, the security will hold under the MCDH assumption. In the standard model, the security relies on the stronger MDDH assumption. We believe these are reasonable assumptions.

About the traitor-tracing functionality, the non-incrimination relies on the CDH assumption, while the traceability of colluders is guaranteed under the DBDH^1-M assumption.

As a consequence, our scheme will achieve the classical security notions of traitor-tracing under the MDDH and DBDH^1-M assumptions.

Public Traceability. Our scheme will provide the new and interesting property of *public traceability*. It however requires stronger assumptions, since more information is available to the adversary (since the tracing capability can be provided to a bad guy.)

About the semantic security of the encryption scheme encryption, in the random-oracle model, the CBDH^2-E assumption is required. The latter is in fact a quite minor extension of the classical CBDH^2 assumption (see proposition 2.) In the standard model, the security relies on the DBDH^2-E assumption. Again, we believe this is a reasonable assumption.

Considering the properties of traitor-tracing, the non-incrimination is captured by the tracing of colluders, which is again proven under the DBDH^1-M assumption.

Conclusion. Finally, our scheme, with public traceability, will essentially require the three new assumptions MDDH for the security of encryption, DBDH^1-M for the traitor-tracing property, and the DBDH^2-E for the public traceability.

4.2 Kiayias-Yung's Scheme

Our construction of 2-user traitor tracing scheme is based on the Kiayias and Yung's scheme [14], which can be seen as a special case of the Boneh and Franklin's scheme [2]. Let us thus first review the KY scheme.

Setup: Given a security parameter $\kappa \in \mathbb{Z}$, the algorithm works as follows:

Step 1: Generate a κ-bit prime q and a group $\mathcal{G}$ of order q. Choose an arbitrary generator $g \in \mathcal{G}$.

Step 2: Pick random elements $a, z \in \mathbb{Z}_q^\star$, and set $Q = g^a$, $Z = g^z$.

Private key of the center: the pair (a, z).

Encryption key: the tuple $\mathsf{pk} = (g, Q, Z)$.

User key: user u_b (for $b \in \{0, 1\}$, since we focus on the two-user case) is associated to a "representation" $k_b = (\alpha_b, \beta_b)$ of g^z with respect to the basis (g, g^a), i.e, the authority selects two vectors (α_0, β_0) and (α_1, β_1) in $\mathbb{Z}_q^2$ so that $\alpha_b + a\beta_b = z \bmod q$ for both $b \in \{0, 1\}$. The two vectors are chosen so that they are linearly independent. The set of all possible keys is

$$\mathcal{K}_{\mathsf{pk}} = \{(\alpha, \beta) | \alpha + a\beta = z \bmod q\}.$$

Encryption Algorithm: The encryption algorithm generates a random $k \in \mathbb{Z}_q$ and outputs a ciphertext (c_1, c_2, d) into $\mathcal{G}^3$: on a plaintext m, assumed to be in the group $\mathcal{G}$, the center computes $C = (c_1 = g^k, c_2 = Q^k, d = m \times Z^k)$. We say that a triple $(c_1, c_2, d) \in \mathcal{G}^3$ is a valid ciphertext if there exits $k \in \mathbb{Z}_q$ such that $c_1 = g^k$ and $c_2 = Q^k$. Otherwise, the ciphertext is invalid.

Decryption Algorithm: On a ciphertext (c_1, c_2, d), user u_b computes:

$$Z^k = c_1^{\alpha} \times c_2^{\beta} \quad \text{and} \quad m = d/Z^k.$$

4.3 Our Construction

We now show how we can use bilinear maps in order to improve this scheme. More precisely, we introduce the notion of public-key traitor tracing with *proxy quantity*. Contrarily to usual public-key traitor tracing schemes, the authority generates for each user a key along with a corresponding *proxy quantity*. The authority keeps in his hands the user's key and gives only to the user the proxy quantity which is enough for decryption. The user's key will be later used for tracing.

Setup: Given a security parameter $\kappa \in \mathbb{Z}$, the algorithm works as follows:

Step 1: Generate a κ-bit prime q, two groups $\mathcal{G}_1$ and $\mathcal{G}_2$ of order q, and an admissible bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$. Choose an arbitrary generator $P \in \mathcal{G}_1$ and set $g = \hat{e}(P, P)$ which is a generator of group $\mathcal{G}_2$.

Step 2: Pick random elements $a, z \in \mathbb{Z}_q^{\star}$, and set $Q = aP$, $Z = g^z$.

Step 3: Choose a function $H : \mathcal{G}_1 \to \mathcal{M}$. The security analysis will view H as either a random oracle or a function in a universal hash function family (using the leftover-hash-lemma [11, 12]).

The message space is $\mathcal{M} = \{0,1\}^{\kappa}$. The ciphertext space is $\mathcal{G}_1^{\star} \times \mathcal{G}_1^{\star} \times \{0,1\}^{\kappa}$. The system parameters are $\mathsf{params} = (q, \mathcal{G}_1, \mathcal{G}_2, \hat{e}, P, H)$.

Private key of the center: the pair (a, z).

Encryption key: the tuple $\mathsf{pk} = (g, Q, Z)$.

User key: user u_b (for $b \in \{0,1\}$) is associated to a "representation" $k_b = (\alpha_b, \beta_b)$ of g^z with respect to the base (g, g^a). The set of all possible keys is

$$\mathcal{K}_{\mathsf{pk}} = \{(\alpha, \beta) | \alpha + a\beta = z \bmod q\}.$$

Remark that the authority generates these keys for each user but *does not give them* to the users. Each user is just given a proxy quantity, as described below.

Proxy Quantity: user u_b (for $b \in \{0,1\}$) receives a proxy quantity $\Pi(k_b) = (\alpha_b, \pi_b = \beta_b P)$. The set of all possible proxy quantities is

$$\Pi_{\mathsf{pk}} = \{(\alpha, \pi = \beta P) | (\alpha, \beta) \in \mathcal{K}_{\mathsf{pk}}\}.$$

Encryption Algorithm: The encryption algorithm generates a random $k \in \mathbb{Z}_q$ and outputs a ciphertext (c_1, c_2, d) into $\mathcal{G}_1 \times \mathcal{G}_1 \times \mathcal{M}$: on a plaintext $m \in \mathcal{M}$, the center computes $C = (c_1 = kP, c_2 = k^2 Q, d = m \oplus H(Z^{k^2}))$. We say that $(c_1, c_2, d) \in \mathcal{G}_1 \times \mathcal{G}_1 \times \mathcal{M}$ is a valid ciphertext if there exits $k \in \mathbb{Z}_q$ such that $c_1 = kP$ and $c_2 = k^2 Q$. Otherwise, the ciphertext is invalid.

Decryption Algorithm: On a ciphertext (c_1, c_2, d), user u_b computes, granted his proxy quantity $\Pi(k_b) = (\alpha_b, \pi_b)$,

$$Z^{k^2} = \hat{e}(\alpha_b c_1, c_1) \cdot \hat{e}(\pi_b, c_2) \quad \text{and} \quad m = d \oplus H(Z^{k^2}).$$

4.4 Rationale

First, one can wonder why we do not encrypt the message by $c_1 = kP$, $c_2 = kQ$ and $d = m \oplus H(Z^k)$, while each user would receive the key $k_b = (\alpha_b, \beta_b)$ (so that $\alpha_b + a\beta_b = z \bmod q$). Such scheme would thus be a simple and natural variation of the KY scheme using bilinear maps. However, as in our above attack against the TSZ scheme, the adversary could take advantage of the bilinear property to combine the secret key and the public information. Actually, any adversary, although it could not produce a new key, could produce and distribute an anonymous decoder $(X = \alpha P - uQ, Y = \beta P + uP)$, in which u could be randomly chosen in $\mathbb{Z}_q$. Then, everyone could recover $Z^k = \hat{e}(X, c_1) \cdot \hat{e}(Y, c_2)$. Because of the random choice of u, the authority cannot trace back the traitor.

In our scheme, we prove that such an adversary cannot exist: users do not have keys of the form (α, β), but proxy quantities only, of the form $(\alpha, \beta P)$. As a consequence, even if two users collude to produce another key (by a linear combination of their keys), they cannot learn the secret key (a, z). We will see that this is crucial to improve the result in the multi-user case.

4.5 Security of the Encryption Scheme

Before considering security properties specific to the traitor tracing functionality, let us first study the encryption scheme. Actually, if we consider the function H as a random oracle, the semantic security of the encryption can be proved under the MCDH problem. If we consider that the function H is randomly chosen in a universal hash function family [11, 12], the semantic security of the encryption is proved under the MDDH problem. The proofs of the following theorems can be found in the full version [6].

Theorem 1. *Let H be seen as a random oracle. The above scheme is semantically secure under the* MCDH *problem.*

Theorem 2. *Let H be a function randomly chosen in a universal hash function family. The above scheme is semantically secure under the* MDDH *problem.*

4.6 Non-incrimination

The main goal of a traitor tracing scheme is to be able to trace a pirate. But a pirate could try to incriminate another user. E.g., using his private information, a pirate could try to produce another proxy quantity and distribute it. We show that this scenario cannot happen. The proof can be found in the appendix.

Theorem 3. *Given the encryption key and a proxy quantity $(\alpha, \pi) \in \Pi_{\mathsf{pk}}$, it is computationally infeasible to construct another proxy quantity in Π_{pk} under the* CDH *problem.*

4.7 Black-Box Traitor Tracing

For practical reasons, it is important not to have to open the pirate decoder in order to trace back the pirate. We thus show that our scheme is black-box traitor tracing against a collusion of the 2 users under the DBDH^1-M problem, by constructing a tracing algorithm. For this security result, we assume that the hash function H is a function randomly chosen in a universal hash function family. The proof can be found in the full version [6], since it is a simpler case than the proof of the Theorem 6 (provided in the appendix.)

Theorem 4. *Let us assume that, given the encryption key* pk *and a proxy quantity* $(\alpha, \pi = \beta P) \in \Pi_{\mathsf{pk}}$*, the adversary* $\mathcal{A}$ *produces a decryption simulator* $\mathcal{S}$ *that decrypts valid ciphertexts, but when given a "randomized" ciphertext of the form* (kP, ak'^2P, d) *with* $k, k' \stackrel{R}{\leftarrow} \mathbb{Z}_q, d \stackrel{R}{\leftarrow} \mathcal{M}$*, it outputs a value different from* $d \oplus H(g^{\alpha k^2 + a\beta k'^2})$ *with probability* ε*. Then the* DBDH^1-M *problem can be solved with an advantage* $\varepsilon/2$*.*

Intuitively, the above theorem shows that a "randomized" and thus invalid ciphertext cannot be distinguished from a regular and valid ciphertext. Therefore, given a black-box access to a decryption simulator $\mathcal{S}$ constructed by one of two users, one can always decide which one of them has built it: one randomly chooses $k, k' \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$ (we suppose that $k \neq k'$), and sets $u_0 = \alpha_0 k^2 + a\beta_0 k'^2$ and $u_1 = \alpha_1 k^2 + a\beta_1 k'^2$. With high probability (greater than $1 - 2/q$), u_0 is different from u_1, which is thus assumed in the following. One then submits the randomized invalid ciphertext (kP, ak'^2P, d). If the output of $\mathcal{S}$ is d/g^{u_0} then one claims that u_0 is the traitor. If the output is d/g^{u_1}, then u_1 is blamed. If the output is none of these two values, one concludes that the two users colluded. Hence the following corollary.

Corollary 1. *The above scheme is black-box traitor tracing against active adversaries.*

4.8 Public Traceability

Let us now turn to the additional and quite interesting property: in order to execute the black-box traitor tracing procedure, the two user-keys (α_0, β_0) and (α_1, β_1) are used. However, the proxy quantities would be enough, and even less: $(\alpha_0 P, \beta_0 P)$ and $(\alpha_1 P, \beta_1 P)$ are sufficient. From $k, k' \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$, one does not really need u_0, u_1, but just g^{u_0} and g^{u_1}:

$$g^{u_0} = \hat{e}(\alpha_0 P, k^2 P) \times \hat{e}(Q, k'^2(\beta_0 P));$$
$$g^{u_1} = \hat{e}(\alpha_1 P, k^2 P) \times \hat{e}(Q, k'^2(\beta_1 P)).$$

This is a quite new and interesting property: one can split the roles of the center. Moreover, the tracing capability can be delegated to several servers in order to speed up the tracing. This delegation does not require any trust in these servers, since the given information does not leak the private key (a, z), nor even any

information to build a decoder (under an additional computational assumption). Furthermore, one can thereafter check whether the incriminated people are the pirates or not.

We now formally state the above security properties in the following theorems whose proofs can be found in the appendix.

Theorem 5. *Let us assume that the tracing information is public, then the encryption scheme is semantically secure: in the random-oracle model, the security relies on the* CBDH^2-E *assumption, while the standard model requires the* DBDH^2-E *assumption.*

Theorem 6. *Let us assume that* $\mathcal{A}$ *is an algorithm which, given the encryption key* pk*, one proxy quantity* $(\alpha, \pi = \beta P)$ *(among the two* $(\alpha_0, \pi_0 = \beta_0 P)$ *and* $(\alpha_1, \pi_1 = \beta_1 P)$ *provided by the center), and the public tracing information* $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$*, can produce a decryption simulator* $\mathcal{S}$ *that decrypts valid ciphertexts, but when given a "randomized" ciphertext of the form* $(kP, ak'^2 P, d)$ *with* $k, k' \stackrel{R}{\leftarrow} \mathbb{Z}_q$*,* $d \stackrel{R}{\leftarrow} \mathcal{M}$*,* $\mathcal{S}$ *outputs a value different than* $d \oplus H(g^{\alpha_0 k^2 + a\beta_0 k'^2})$ *with probability* ε*. Then the* DBDH^1-M *problem can be solved with advantage* $\varepsilon/2$*.*

5 The Multi-user Case

5.1 Description

Let $C = \{\omega_1, \ldots, \omega_N\}$ be an $(N, c, \ell, \varepsilon)$-collusion-secure code over the alphabet $\{0, 1\}$ with ℓ-long codewords, that allows collusions of up to c users and has a tracing algorithm that succeeds with probability $1 - \varepsilon$ (see [4] for more details). The multi-user case (ℓ-key system) is simply ℓ-instantiations of the 2-user public-key 1-traitor tracing scheme with proxy quantities. We indeed build such an ℓ-key system using an $(N, c, \ell, \varepsilon)$-collusion-secure code C as a combination of ℓ 2-user systems S_1, S_2, ..., S_ℓ:

Setup: Given the security parameters k, c and ε:

- **Step 1:** Generate a k-bit prime q, two groups $\mathcal{G}_1$, $\mathcal{G}_2$ of order q, and an admissible bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$. Choose an arbitrary generator $P \in \mathcal{G}_1$.
- **Step 2:** Generate an $(N, c, \ell, \varepsilon)$-collusion-secure code $C = \{\omega_1, .., \omega_N\}$.
- **Step 3:** Pick random elements $a, z_j \in \mathbb{Z}_q^\star$, and set $Q = aP$, $Z_j = g^{z_j}$, for $j = 1, \ldots, \ell$.
- **Step 4:** Choose a function $H : \mathcal{G}_1 \to \mathcal{M}$.

The system parameters are $\mathsf{params} = (q, \mathcal{G}_1, \mathcal{G}_2, \hat{e}, P, H)$. These parameters are common for all 2-user systems S_1, S_2, ..., S_ℓ.

Private key of the center: the element a, and the tuple $(z_j)_{j=1,\ldots,\ell}$.

Encryption key: this is the combination of the encryption keys from the ℓ 2-user schemes: $\mathsf{pk} = (g, Q, \{Z_j = g^{z_j}\}_{j=1,\ldots,\ell})$.

User key: user u_i (for $i \in \mathbb{Z}_N$) is associated to a codeword ω_i in C and the corresponding "representation" $(\alpha_{\omega_{i,j},j}, \beta_{\omega_{i,j},j})$ of g^{z_j} with respect to the basis

(g, g^a), where $\omega_{i,j}$ is the j-th bit of the codeword ω_i. Recall that $(\alpha_{b,j}, \beta_{b,j})$ is a "representation"of g^{z_j} with respect to the base (g, g^a). Again, this user key is *not* given to the user, but only the proxy quantity.

Proxy Quantity: user u_i (for $i \in \mathbb{Z}_N$) is given the proxy quantity $\Pi_i = (\Pi_{\omega_{i,1},1}, \ldots, \Pi_{\omega_{i,l},\ell})$. More precisely, for $j = 1, \ldots, \ell$,

$$\Pi_{\omega_{i,j},j} = (\alpha_{\omega_{i,j},j}, \pi_{\omega_{i,j},j} = \beta_{\omega_{i,j},j} P).$$

Encryption algorithm: The plaintext space of the ℓ-key system is $\mathcal{M}^\ell$. On input $(m_1, \ldots, m_\ell)$, the encryption algorithm uses a random $k \in \mathbb{Z}_q$ and outputs the ciphertext $(c_1, c_2, d_1, \ldots, d_\ell)$ into $\mathcal{G}_1 \times \mathcal{G}_1 \times \mathcal{G}_2^\ell$, where: $c_1 = k \times P$, $c_2 = k^2 \times aP$ and $d_j = m_j \oplus H(Z_j^{k^2})$.

Decryption Algorithm: On the ciphertext $(c_1, c_2, d_1, \ldots, d_\ell)$, user u_i computes, granted his proxy quantity, $Z_j^{k^2} = \hat{e}(\alpha_{\omega_{i,j},j} c_1, c_1) \times \hat{e}(\pi_{\omega_{i,j},j}, c_2)$ and then $m_j = d_j \oplus H(Z_j^{k^2})$.

For the security analysis, one could use the following assumption, from [14]: the *threshold assumption* says that a pirate-decoder that just returns correctly a fraction p of a plaintext of length λ where $1-p$ is a non-negligible function in λ, is useless. However, as already mentioned in [14], by employing an all-or-nothing transform [18, 5], this assumption is not necessary.

Proposition 3. *The collusion of the users in the $(\ell - 1)$ 2-user systems of ℓ 2-user systems does not affect the security of the remained 2-user system.*

This proposition which proof can be found in the full version [6], combined with the fact that C is an $(N, c, \ell, \varepsilon)$-collusion-secure code, leads to following corollary:

Corollary 2. *The above scheme is a N-user, c-traitor tracing scheme.*

About the public traceability, with the public information, anybody can recover the codeword associated to the pirate decoder, the interactive and thus costly phasis. However, classical collusion-secure codes do not allow to publicly trace back to a guilty, but this is an off-line prodecure, which still must be performed by a trusted authority.

5.2 Comparison with the Kiayias-Yung's Scheme

In the KY scheme, the ciphertext rate is 3, while ours is asymptotically 1. One could wonder why we could use the above construction, while they could not.

The reason is that in our 2-user scheme, even the collusion of the 2 users does not leak any information about a. In the KY 2-user scheme, such a collusion immediately reveals a: in the multi-user case, if one uses the same a for the ℓ 2-user schemes, the collusion of two users could leak this value a and then all the values z_i, which would easily lead to an anonymous pirate decoder. As a consequence, they have to use distinct a's in each 2-user scheme instance, while in our scheme, a common a is possible.

6 Conclusion

We thus proposed a scheme which improves the Kiayias and Yung's scheme in various ways: first, the transmission rates are reduced near optimality; and we introduce the quite interesting functionality of *public traceability*. The full feature of public traceability in the multi-user case, which would lead to a guilty, is however an open problem.

Acknowledgement

The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

References

1. D. Boneh and X. Boyen. Short signatures without random oracles. In *Adv. in Cryptology – Proceedings of Eurocrypt 2004*, volume LNCS 3152, pages 56–73. Springer-Verlag, 2004.
2. D. Boneh and M. Franklin. An efficient public key traitor tracing scheme. In M.Wiener, editor, *Adv. in Cryptology – Proceedings of Crypto 1999*, volume LNCS 1666, pages 338–353. Springer-Verlag, 1999.
3. D. Boneh and M. Franklin. Identity-based Encryption from the Weil Pairing. In J. Kilian, editor, *Adv. in Cryptology – Proceedings of Crypto '01*, LNCS 2139, pages 213–229. Springer-Verlag, Berlin, 2001.
4. D. Boneh and J. Shaw. Collusion secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.
5. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-Resilient Functions and All-Or-Nothing Transforms. In *Eurocrypt '00*, LNCS 1807, pages 453–469. Springer-Verlag, Berlin, 2000.
6. H. Chabanne, D. H. Phan, and D. Pointcheval. Public Traceability in Traitor Tracing Schemes. In *Eurocrypt '05*, LNCS. Springer-Verlag, Berlin, 2005. Full version available from `http://www.di.ens.fr/users/pointche`.
7. B. Chor, A. Fiat and M. Naor. Tracing traitor. In Y. Desmedt, editor, *Adv. in Cryptology – Proceedings of Crypto 1994*, volume LNCS 839, pages 257–270. Springer-Verlag, 1994.
8. B. Chor, A. Fiat, M. Naor and B. Pinkas. Tracing traitor. *IEEE Transactions on Information Theory*, 46(3): 893–910, 2000.
9. Y. Dodis and N. Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Y.G.Desmedt, editor, *Proceedings of PKC 2003*, volume LNCS 2567, pages 100–115. Springer-Verlag, 2003.
10. E. Gagni, J. Staddon, and Y.L. Yin. Efficient methods for integrating traceability and broadcast encryption. In M.Wiener, editor, *Adv. in Cryptology – Proceedings of Crypto 1999*, volume LNCS 1666, pages 372–387. Springer-Verlag, 1999.
11. J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A Pseudorandom Generator from any One-Way Function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999.
12. I. Impagliazzo, L. Levin, and M. Luby. Pseudo-Random Generation from One-Way Functions. In *Proc. of the 21st STOC*, pages 12–24. ACM Press, New York, 1989.

13. A. Joux. A One-Round Protocol for Tripartite Diffie-Hellman. In *Algorithmic Number Theory Symposium (ANTS IV)*, LNCS 1838, pages 385–394. Springer-Verlag, Berlin, 2000.
14. A. Kiayias and M. Yung. Traitor tracing with constant transmission rate. In L. Knudsen, editor, *Adv. in Cryptology – Proceedings of Eurocrypt 2002*, volume LNCS 2332, pages 450–465. Springer-Verlag, 2002.
15. A. Kiayias and M. Yung. Breaking and repairing asymmetric public-key traitor tracing. In J. Feigenbaum, editor, *ACM Workshop in Digital Rights Management – DRM 2002*, volume LNCS 2696, pages 32–50. Springer, 2003.
16. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing scheme. *IEICE Trans. Fundamentals*, E85-A(2), 2002.
17. M. Naor and B. Pinkas. Efficient Trace and Revoke Schemes. In *Proc. of Financial Crypto '2000*, volume LNCS 1692, pages 1–20. Springer-Verlag, 2000.
18. R. Rivest. All-or-Nothing Encryption and the Package Transform. In *Proc. of the 4th FSE*, LNCS 1267. Springer-Verlag, Berlin, 1997.
19. V.D. To and R. Safavi-Naini. Linear code implies public-key traitor tracing with revocation. In H. Wang, editor, *Proceedings of ACISP 2003*, volume LNCS 3108, pages 24–35. Springer-Verlag, 2003.
20. V.D. To, R. Safavi-Naini, and F. Zhang. New traitor tracing schemes using bilinear map. In *Proceedings of the 2003 ACM Workshop on Digital Rights Management*, pages 67–76, 2003.

A Proof for the Non-incrimination and Traitor-Tracing

A.1 Proof of the Theorem 3

Let us assume that, given a proxy quantity $(\alpha, \pi) \in \Pi_{\mathsf{pk}}$, $\mathcal{A}$ can construct another proxy quantity. We then construct an algorithm $\mathcal{B}$ that solves the inverse Diffie-Hellman problem, which is well-know to be equivalent to the CDH problem. Algorithm $\mathcal{B}$ is given as input a random instance $(P, A = aP)$ of the inverse Diffie-Hellman problem. Let $B = a^{-1}P$ be the solution, that $\mathcal{B}$ finds as follows:

Setup: $\mathcal{B}$ randomly chooses $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$ and $\pi \stackrel{R}{\leftarrow} \mathcal{G}_1$), and then computes $Z = \hat{e}(P, \alpha P)\hat{e}(A, \pi)$. Finally, $\mathcal{B}$ sets the public key $\mathsf{pk} = (g, A, Z)$. It then sends pk to $\mathcal{A}$ as well as a proxy quantity (α, π).
Attack: $\mathcal{A}$ outputs another proxy $(\tilde{\alpha}, \tilde{\pi})$.
Break: $\mathcal{B}$ computes $c = \tilde{\alpha} - \alpha$ and outputs $B = (c^{-1} \bmod q)(\pi - \tilde{\pi})$.

We see that, since $(\tilde{\alpha}, \tilde{\pi})$ is a new proxy, for any ciphertext (c_1, c_2, d),

$$\begin{aligned}
&\hat{e}(\alpha c_1, c_1) \times \hat{e}(\pi, c_2) = \hat{e}(\tilde{\alpha} c_1, c_1) \times \hat{e}(\tilde{\pi}, c_2) \\
\iff &\hat{e}(\alpha k P, kP) \times \hat{e}(\pi, ak^2 P) = \hat{e}(\tilde{\alpha} kP, kP) \times \hat{e}(\tilde{\pi}, ak^2 P) \\
\iff &\hat{e}(\alpha P, P) \times \hat{e}(\pi, aP) = \hat{e}(\tilde{\alpha} P, P) \times \hat{e}(\tilde{\pi}, aP) \\
\iff &\hat{e}((\tilde{\alpha} - \alpha)P, P) = \hat{e}(\pi - \tilde{\pi}, aP).
\end{aligned}$$

From the fact that $(\alpha, \pi), (\tilde{\alpha}, \tilde{\pi}) \in \Pi_{\mathsf{pk}}$, we get that π and $\tilde{\pi}$ are in the group $\mathcal{G}_1$ generated by P. Therefore, $\pi - \tilde{\pi} = bP$ for some b. We then have $c = \tilde{\alpha} - \alpha = ab$ and thus:

$$B = (c^{-1} \bmod q)(\pi - \tilde{\pi}) = (ab)^{-1} \times bP = a^{-1}P. \qquad \square$$

B Proofs for the Public Traceability

B.1 Proof of the Theorem 5

We focus on the case where the function H is randomly chosen in a universal hash function family. The case where H is a random oracle is similar to the proof of the Theorem 1.

Let us assume that the scheme is not semantically secure against passive adversaries. Then there is an $\mathsf{IND\text{-}CPA}$ adversary $\mathcal{A}$ that, given the public key pk and the tracing information $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$, can break the scheme with advantage ε. We can then construct an algorithm $\mathcal{B}$ that solves the $\mathsf{DBDH}^2\text{-}\mathsf{E}$ problem. Algorithm $\mathcal{B}$ is given as input a random $\mathsf{DBDH}^2\text{-}\mathsf{E}$ instance $(P, A = aP, B = kP, C = zP, D = ak^2P, U)$ from either the distribution in which U is the $\mathsf{CBDH}^2\text{-}\mathsf{E}$ solution, or the distribution in which U is a random element in $\mathcal{G}_2$. The algorithm $\mathcal{B}$ runs as follows:

Setup: $\mathcal{B}$ sets the public key $\mathsf{pk} = (g, Q = A, Z = g^z = \hat{e}(C, P))$. $\mathcal{B}$ randomly chooses β_0, β_1 and computes:

$$\alpha_0 P = zP - \beta_0 Q \qquad \alpha_1 P = zP - \beta_1 Q.$$

It sends pk, along with $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$ to $\mathcal{A}$.

Challenger: $\mathcal{A}$ outputs two message m_0, m_1 on which it wishes to be challenged. $\mathcal{B}$ picks a random element $b \in \{0, 1\}$ and gives $(A, D, d = m_b \oplus H(U))$ as the challenge to $\mathcal{A}$.

Guess: Algorithm $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. At this point, $\mathcal{B}$ returns 1 if $b = b'$ and 0 otherwise.

Observe that if U is the $\mathsf{CBDH}^2\text{-}\mathsf{E}$ solution, then the challenge ciphertext is an encryption of m_b. Otherwise, since H is randomly chosen from a universal hash function family, the challenge is the ciphertext of a random message, hence $b = b'$ holds with probability $1/2$. By a standard argument, the adversary $\mathcal{B}$ has an advantage of $\varepsilon/2$ in deciding $\mathsf{DBDH}^2\text{-}\mathsf{E}$. □

B.2 Proof of the Theorem 6

From such an adversary $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that breaks the $\mathsf{DBDH}^1\text{-}\mathsf{M}$ problem: Algorithm $\mathcal{B}$ is given as input the $\mathsf{DBDH}^1\text{-}\mathsf{M}$ parameters $(\mathcal{G}_1, \mathcal{G}_2, \hat{e})$ together with a random instance $(P, A = aP, B = kP, X)$ (for $a, k \overset{R}{\leftarrow} \mathbb{Z}_q^\star$, $X \overset{R}{\leftarrow} \mathcal{G}_1$). Algorithm $\mathcal{B}$ decides whether $X = ak^2P$ by interacting with $\mathcal{A}$

Setup: $\mathcal{B}$ randomly chooses $\alpha_0, \beta_0, \beta_1 \overset{R}{\leftarrow} \mathbb{Z}_q^\star$ and computes:

$$zP = \alpha_0 P + \beta_0 A \qquad \alpha_1 P = zP - \beta_1 A \qquad \pi_0 = \beta_0 P \qquad Z = \hat{e}(P, zP).$$

$\mathcal{B}$ sets $Q = A$, $\mathsf{pk} = (g, Q, Z)$ and a proxy (α_0, π_0), as well as the public tracing information $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$. The proxy (α_0, π_0) can be considered as randomly chosen in the set Π_{pk}. Finally, $\mathcal{B}$ gives pk, the proxy (α_0, π_0) and the public tracing information $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$ to $\mathcal{A}$.

Ciphertext: $\mathcal{B}$ randomly chooses $d \in \mathcal{M}$ and builds a ciphertext $(c_1 = B, c_2 = X, d)$. $\mathcal{B}$ sends it to $\mathcal{A}$. Because of the random choice of (B, X, d) and the random choice of the hash function H in a universal hash function family, the challenge (c_1, c_2, d) is a random ciphertext.

Break: If algorithm $\mathcal{A}$ returns $d \oplus H(\hat{e}(\alpha_0 c_1, c_1) \cdot \hat{e}(\pi_0, X))$, $\mathcal{B}$ outputs randomly yes or no. Otherwise $\mathcal{B}$ output no (X is certainly not ak^2P).

Note that when $X = ak^2P$, the ciphertext (kP, X, d) is a random valid ciphertext and the algorithm $\mathcal{A}$ outputs correctly the plaintext $m = d \oplus H(\hat{e}(\alpha_0 c_1, c_1) \cdot \hat{e}(\pi_0, X))$. In this case, the algorithm $\mathcal{B}$ outputs randomly yes or no and the probability that $\mathcal{B}$ gives a correct guess is $1/2$.

When $X \neq ak^2P$, the ciphertext (kP, X, d) is a random invalid ciphertext. Since the decoder behaves differently for invalid ciphertext with probability ε, $\mathcal{A}$ outputs differently than the expected plaintext with probability ε. In such a case, $\mathcal{B}$ answers correctly that X is not equal to ak^2P. In the case $\mathcal{A}$ outputs the expected plaintext (which happens with probability less than $1 - \varepsilon$), $\mathcal{B}$ answers randomly yes or no. Therefore, when $X \neq ak^2P$, the probability that $\mathcal{B}$ gives a correct guess is $\varepsilon + (1 - \varepsilon) \times 1/2 = 1/2 + \varepsilon/2$.

Combining the two above cases, we easily see that $\mathcal{B}$ can solve the DBDH^1-M problem with advantage $\varepsilon/2$. □

One-Way Chain Based Broadcast Encryption Schemes*

Nam-Su Jho[1], Jung Yeon Hwang[2,**], Jung Hee Cheon[1], Myung-Hwan Kim[1], Dong Hoon Lee[2], and Eun Sun Yoo[1]

[1] ISaC and Department of Mathematical Sciences, Seoul National University, Seoul 151-747, Korea
{drake, jhcheon, mhkim, eunsun}@math.snu.ac.kr
[2] Graduate School of Information Security CIST, Korea University, Seoul 136-701, Korea
{videmot, donghlee}@korea.ac.kr

Abstract. We propose a new broadcast encryption scheme based on the idea of 'one key per each punctured interval'. Let r be the number of revoked users. In our scheme with p-punctured c-intervals, the transmission overhead is roughly $\frac{r}{p+1}$ as r grows. Our scheme is very flexible with two parameters p and c. We may take p as large as possible if a user device allows a large key storage, and set c as small as possible if the storage size and the computing power is limited. As variants of the proposed scheme, we further study a combination of a one-way chain and a hierarchical ring. This combination provides a fine-grained trade-off between user storage and transmission overhead. As one specific instance, the combination includes the subset difference (SD) scheme which is considered the most efficient one in the literature.

1 Introduction

Broadcast encryption (BE) is a cryptographic method for a center to efficiently broadcast digital contents to a large set of users so that only non-revoked users can decrypt the contents. In broadcast encryption, the center distributes to each user u the set K_u of keys, called the *user key set* of u, in the system setup stage. We assume that the user keys are not updated afterwards, that is, user keys are *stateless*. A *session* is a time interval during which only one encrypted message (digital contents) is broadcasted. The *session key*, say SK, is the key used to encrypt the contents of the session. In order to broadcast a message M, the center encrypts M using the session key SK and broadcasts the encrypted message together with a *header*, which contains encryptions of SK and the information for non-revoked users to recover SK. In other words, the center broadcasts

$$\langle \text{header} ; E_{SK}(M) \rangle,$$

* This work was supported by Samsung Advanced Institute of Technology.
** Corresponding author.

R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 559–574, 2005.

where $E_{SK}(M)$ is a symmetric encryption of M by SK. Then, every non-revoked user u computes $F(K_u, \text{header}) = SK$ and decrypts $E_{SK}(M)$ with SK, where F is a predefined algorithm. But for any revoked user v, $F(K_v, \text{header})$ should not render SK. Furthermore, there should be no polynomial time algorithm that outputs SK even with all the revoked user keys and the header as input.

The length of the header, the computing time of F and the size of a user key are called the *transmission overhead*, the *computation cost* and the *storage size*, respectively. The main issue of broadcast encryption is to minimize the transmission overhead with practical computation cost and storage size.

The notion of broadcast encryption was first introduced by Berkovits [2] in 1991 using polynomial interpolation and vector based secret sharing. Fiat and Naor [7] in 1993 suggested a formal definition of broadcast encryption and proposed a systematic method of broadcast encryption. The polynomial interpolation method was improved by Naor and Pinkas [14] in 2000 to allow multiple usage. The first practical broadcast encryption scheme was proposed in 2001 by Naor *et al.* [13], called the *Subset Difference* (SD) method. This was improved by Halevi and Shamir [11] in 2002 by adopting the notion of layers and thereby the improved scheme is called the *Layered Subset Difference* (LSD) method. Both SD and LSD are based on tree structure and they are the best known broadcast schemes up to now. To be more precise, let N be the total number of users and r be the number of revoked users. The SD scheme requires $2r$ transmission overhead and $O(\log^2 N)$ storage size for each user. The computation cost is only $O(\log N)$ computations of one-way permutations. The LSD scheme reduces the storage size to $O(\log^{3/2} N)$ while keeping the computation cost same. But the transmission overhead increases to $4r$ in LSD. For other interesting recent articles on broadcast encryption, we refer the readers [3, 8].

Our Contribution. In this paper, we propose a new broadcast encryption scheme based on the idea of "one key per each punctured interval". It has been a general belief that at least one key per each revoked user should be included in the overhead and hence r seems to be the lower bound of the transmission overhead in any broadcast encryption scheme with reasonable computation cost and storage size. In our scheme with p-punctured c-intervals, however, the transmission overhead is about $\frac{r}{p+1} + \frac{N-r}{c}$ which breaks the barrier of r, for the first time under our knowledge if r is not too small, even when $p = 1$, where c is a predetermined constant and r is not too small. Although we set $c = 100$ or 1000 for comparison purpose here, we can choose any c that is suitable for other purposes. The computation cost is very cheap with only $c - 1$ computations of one-way permutations. The storage size is $O(c^{p+1})$, which is practical for most user devices if p is small. Our scheme is very flexible with two parameters p and c. If a user device allows a large key storage like set-top boxes and mobile devices, then we may take p as large as possible to reduce the transmission overhead, which is much more expensive. If a user device has limited storage and computing power like smart cards and sensors, then we may set c as small as possible. Another remarkable feature of our scheme is that it does not have to

preset the total number of users - any number of additional users can join at any time, which is not possible in tree based schemes.

Our idea is to put all the users on a straight line and divide the line into subintervals of length at most c beginning and ending with non-revoked users containing p or less revoked users in between. Then, to each of such intervals, the center assigns just one key, which can be derived by all non-revoked users in the interval, for decrypting the session key. For practical purpose, we introduce a layered variance of our scheme to improve the efficiency for very small r. Compared with SD and LSD, the variance beats them in the transmission overhead. As for the the storage size, ours is better than SD when $p = 0$ and a little bit worse when $p \geq 1$.

Furthermore, we study a combination of a one-way chain and a hierarchical ring to provide a trade-off between transmission overhead and keys storage size per user. As a building block we first present a simple ring structure of which transmission overhead is proportional to the number r of revoked users while each user stores N keys. Then, by transforming the simple ring structure to a hierarchical ring one recursively, we extend the basic scheme to more generalized revocation schemes. Interestingly, our specific example of one extreme side is structurally equivalent to SD [13] with $1+\frac{1}{2}(\log^2 n+\log n)$ storage size and $2r-1$ transmission overhead.

Organization. The rest of this paper is organized as follows: In section 2, we propose revocation schemes with p-punctured intervals together with efficiency and security analysis and introduce layers to our scheme. In Section 3, we propose two revocation schemes based on a ring structure: The first one is a basic scheme with r transmission overhead and the second one is an extension to reduce the transmission overhead below r. We generalize the schemes of a simple ring structure to a *hierarchical* ring structure. In Section 4, we compare our schemes with SD and LSD schemes, and give concluding remarks in Section 5.

2 The Punctured Interval Scheme π

A broadcast encryption scheme involves the center (the message sender) and the set of users (the receivers). Our revocation method is based on a so-called *Subset-Cover* framework proposed by Naor et. al [13]. It consists of three phases as follows:

- The initialization phase: the center provides each user with his/her secret keys that will be used when computing his/her partition key K.
- The broadcast phase: when the center wants to transmit a message M, it partitions the set of all privileged users into disjoint subsets $S_1, ..., S_m$ and computes the partition key K_i corresponding to each S_i, and then broadcasts

$$\langle \text{info}_1, \text{info}_2, \ldots, \text{info}_m ; \mathcal{E}_{K_1}(SK), \mathcal{E}_{K_2}(SK), \ldots, \mathcal{E}_{K_m}(SK) ; \ E_{SK}(M) \rangle,$$

  where info_i is the information on S_i, SK is the session key, and $\mathcal{E}, E$ are symmetric encryption functions. The info_i's and $\mathcal{E}_{K_i}(SK)$'s together form the header.

- The decryption phase : each user first finds the partition S_i where he/she belongs from info$_i$'s, computes the partition key K_i using his/her secret keys, and then decrypts SK and M in order.

2.1 Punctured Intervals

Assume that L be a straight line with N dots (users) on it, where N is the number of total users. In our scheme, each user is indexed by an integer $k \in [1, N]$ and he/she is represented by the k-th dot, denoted by u_k, in the line L. Let $p \geq 0$ and $c > 0$ be integers. By a *p-punctured c-interval* we mean a subset of L which contains c or less consecutive users starting from and ending at non-revoked users and containing p or less revoked users. Let $\mathcal{S}_{(p\,;\,c)}$ be the set of all p-punctured c-intervals.

In each session, the p-punctured c-intervals are to be determined under the following rule :

- The first p-punctured c-interval starts from the leftmost non-revoked user, and each of the following starts from the first non-revoked user after the last non-revoked user of the previous.
- Each p-punctured c-interval contains the maximal possible number of users.

Fig.1 illustrates how to make p-punctured c-intervals with an example when $p = 1, c = 6$:

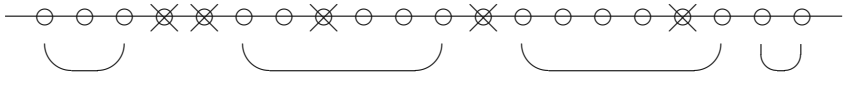

Fig. 1. 1-punctured 6-intervals

The p-punctured c-interval starting from u_i and ending at u_j with $u_{x_1}, \ldots, u_{x_q}$ revoked users is denoted by $P_{i,j;x_1,\ldots,x_q}$ or $P_{i,j;X}$ in short for $X = \{x_1, \ldots, x_q\}$, where $1 \leq j - i + 1 \leq c$, $0 \leq q \leq p$, and $i < x_1 < \cdots < x_q < j$ if there are revoked users.

2.2 Punctured Interval Scheme $(p\,;c)$-π

In this subsection, we propose the punctured interval broadcast encryption scheme $(p\,;c)$-π (PI - Punctured Interval). We assign just one key to each p-punctured c-interval, which can be easily derived by all non-revoked users in that interval, and construct key chains using one-way permutations in order to reduce the storage size.

Key Generation. Let $h_t : \{0,1\}^\ell \to \{0,1\}^\ell$ be one-way permutations for $t = 0, 1, \ldots, p$, where ℓ is the key length. To assign one key to each p-punctured

interval, we randomly choose N keys $K_{1,1}$, $K_{2,2}$, ..., $K_{N,N}$ to be given to $u_1, \ldots, u_N$, respectively. From each $K_{i,i}$ the center constructs the one-way key chains under the following rule : For any possible p-punctured c-interval P starting from u_i given,

- The one-way key chain consists only of the keys of all non-revoked users in P. There are no keys of the revoked users in the chain.
- For any non-revoked user $u_k \in P$, if the next user $u_{k+1} \in P$ is also non-revoked, then just apply h_0 to the key of u_k to obtain the key of u_{k+1}.
- If the next t users are revoked and the user $u_{k+t+1} \in P$ is non-revoked, then apply h_t to the key of u_k to obtain the key of u_{k+t+1}, where $1 \le t \le p$.

The following example illustrates how to construct the key chain of a given punctured interval (with $p = 10, c = 20$) :

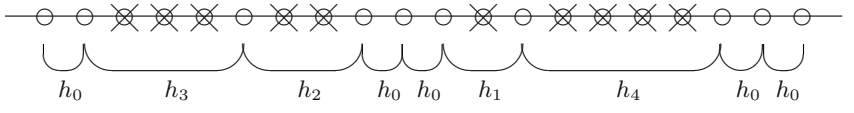

Fig. 2. The key chain of a 10-punctured 20-interval

In the key chain of $P = P_{i,j;x_1,\ldots,x_q}$, the key of a non-revoked user $u_k \in P$ is denoted by $K_{i,k;x_1,\ldots,x_t}$, where $i < x_1 < \cdots < x_t < k < x_{t+1} < \cdots < x_q$ and $0 \le t \le q \le p$. For examples,

$$K_{5,11} = h_0^6(K_{5,5})\,;\ K_{5,11;7} = h_0^3 h_1 h_0(K_{5,5})\,;\ K_{4,11;5,6,7,9,10} = h_2 h_3(K_{4,4})\,;$$
$$K_{3,11;4,5,7,8} = h_0^2 h_2^2(K_{3,3})\,;\ K_{3,11;4,5,6,7,9} = h_0 h_1 h_4(K_{3,3})\,;\ \ldots.$$

The center assigns these keys to users so that the user u_k receives $K_{k,k}$ and all possible $K_{i,k;x_1,\ldots,x_t}$'s, where $i < x_1 < x_2 < \cdots < x_t < k$ with $0 \le t \le p$ and $2 \le k - i + 1 \le c$. Fig. 3 describes the key assignment in the scheme $(3\,;5)$-π for u_5 :

Encryption. For each session, the center divides L into disjoint p-punctured c-intervals $P_1, \ldots, P_m \in \mathcal{S}_{(p\,;\,c)}$, whose union covers all the non-revoked users. Let $P = P_{i,j;x_1,\ldots,x_q}$ be one of P_μ's. The last key $K_{i,j;x_1,\ldots,x_q}$ of the key chain corresponding to P is called the *interval key* of P. Let's denote the interval key of P_μ by K_μ for each $\mu = 1, 2 \ldots, m$, just for convenience. Then the center broadcasts :

$$\langle\, \mathrm{info}_1, \mathrm{info}_2, \ldots, \mathrm{info}_m\,;\, \mathcal{E}_{K_1}(SK), \mathcal{E}_{K_2}(SK), \ldots, \mathcal{E}_{K_m}(SK)\,;\ E_{SK}(M)\,\rangle,$$

where info_μ is information on P_μ, the μ-th interval starting from u_{i_μ} and ending at u_{j_μ} with q_μ revoked users. For each μ, info_μ consists of $i_\mu, \ell_\mu, \ell_{\mu,1}, \ldots, \ell_{\mu,q_\mu}$, where $\ell_\mu = j_\mu - i_\mu + 1$ and $\ell_{\mu,1}, \ldots, \ell_{\mu,q_\mu}$ are the distances from u_{i_μ} to the

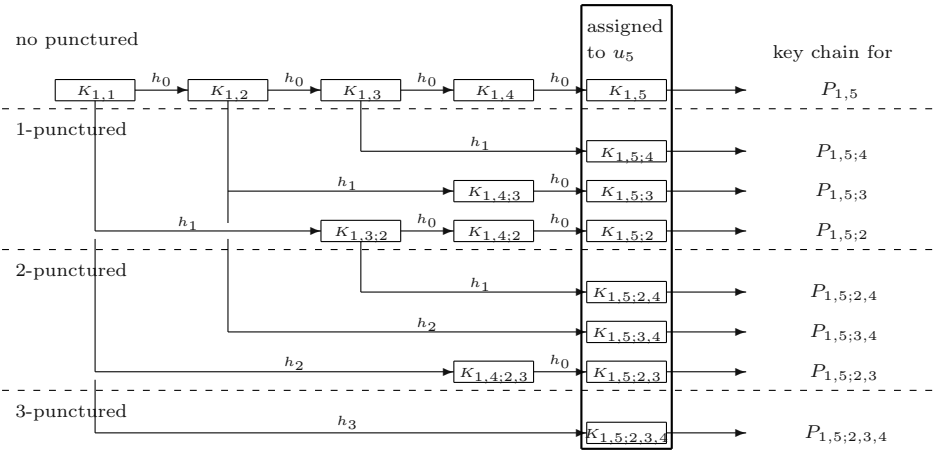

Fig. 3. One-way key chains starting from $K_{1,1}$, where $c = 5$

first, ..., to the last revoked users of P_μ, respectively. The starting position i_μ can be represented by $\log N$ bits and the ℓ's are at most $\log c$ bits. So the size of all info's is $m(\log N + p \log c)$, which will be ignored when computing the transmission overhead because it is negligible compared to the size of all $\mathcal{E}_K(SK)$'s.

Decryption. Receiving the encrypted message, each non-revoked user u_k first locates the punctured interval that he/she belongs using the info's. Let the punctured interval be $P_{i,j;x_1,\ldots,x_q}$, where $i \le k \le j, k \ne x_1, \ldots, x_q$. Then u_k can find $K_{i,j;x_1,\ldots,x_q}$ as follows:

- Find t for which $x_t < k < x_{t+1}$, where $0 \le t \le q$. Here, $t = 0$ and $t = q$ mean that there is no revoked user before and after u_k, respectively.
- Choose $K_{i,k;x_1,\ldots,x_t}$ from the assigned user keys.
- Starting from $K_{i,k;x_1,\ldots,x_t}$, apply one-way permutation h_i's under the rule described in Key Generation until the second subscript reaches to j.
- The resulting key is then $K_{i,j;x_1,\ldots,x_q}$.

With this, u_k decrypts $E_{K_{i,j;x_1,\ldots,x_q}}(SK)$ and $E_{SK}(M)$ to obtain the session key SK and the message M, respectively, in order.

2.3 Efficiency

We analyze efficiency - the transmission overhead (TO), the computation cost (CC) and the storage size (SS) - of the scheme $(p\,;c)$-π.

The transmission overhead of the scheme $(p\,;c)$-π is

$$\mathrm{TO}_{(p\,;c)}(N,r) = \left\lfloor \frac{r}{p+1} \right\rfloor + \left\lceil \frac{N-(p+2)\lfloor r/(p+1)\rfloor}{c} \right\rceil, \tag{1}$$

where N and r are the total number and revoked users, respectively. Especially,

$$\mathrm{TO}_{(1\,;\,c)}(N,r) = \lfloor r/2 \rfloor + \left\lceil \frac{N - 3\lfloor r/2 \rfloor}{c} \right\rceil .$$

This occurs when $\circ\times\times$ is repeated from the leftmost user and then the remaining privileged users are on the right, where $\circ$ and $\times$ denote a privileged user and a revoked user, respectively. It is not hard to prove (1), but we omit the proof because it is long and tedious.

It is trivial that the computation cost $CC_{(p;c)}$ is at most $c-1$ computations of one-way permutations, which is independent of N and r. The storage size of each user can be easily computed as follows:

$$SS_{(p\,;\,c)} = \sum_{k=0}^{p} \left(\frac{1}{(k+1)!} \prod_{i=1}^{k+1} (c-i) \right) + 1,$$

which is also independent of N and r.

2.4 Security

Note that even a non-revoked user cannot compute the interval keys of the other punctured intervals. Those who do not belong to any punctured interval are the revoked ones and they can never access to the session key. Neither those revoked users who belong to punctured intervals can access to their interval keys because they cannot invert the one-way permutations.

The only way to compute the interval key $K_{i,j;x_1,\ldots,x_q}$ of $P_{i,j;x_1,\ldots,x_q}$ is to obtain one of the keys in the key chain. However, no revoked user is assigned a key in the key chain and hence they cannot compute the interval key even though they all collude. Furthermore, the interval keys of previous sessions when the user was not revoked do not help at all in the present session, in which he/she is revoked, because the revocation of him/her results in a totally new key chain.

2.5 Layered Punctured Interval Scheme

The scheme $(p\,;c)$-π is less efficient than SD when r is small. This is mainly because of long intervals consisting of non-revoked users which require several keys while covering no revoked users at all. To deal with this case, we introduce another set of user keys, each of which covers a long interval. To reduce the number of keys, we restrict the starting points of long intervals to some special *nodes* (users) on the line such that the distance between every neighboring nodes, called *node-distance* is c. This process can be repeated by $d-1$ more times taking special nodes with node distances are $c^2, c^3, \ldots, c^{d-1}$ or c^d, respectively, for a positive integer d. We call this scheme by *d-layered p-punctured c-interval* scheme or the $(p\,;c)$-π_d scheme.

Layered Structure. As in the $(p\,;c)$-π scheme, the set of all N users are arranged on a long line L. Given a positive integer d ($< \log_c N - 1$), we consider d layers above the line L. The first layer L_1 consists of $N_1 = \lceil \frac{N}{c} \rceil - 1$ users

$u_1, u_{c+1}, \ldots, u_{(N_1-1)c+1}$. Inductively, the t-th layer L_t consists of $N_t = \lceil \frac{N_{t-1}}{c} \rceil - 1$ users $u_1, u_{c^t+1}, \ldots, u_{(N_t-1)c^t+1}$ for $1 < t \leq d$. We define layered intervals of length c^t in the layer L_t by

$$LP_i^{(t)} = \{u_k | (i-1)c^t + 1 \leq k \leq ic^t\}. \tag{2}$$

Key Assignment. First, the center assigns a random key $LK_i^{(t)}$ to $LP_i^{(t)}$ for each i and gives it to all members of $LP_i^{(t)}$. Next, it constructs a one-way key chain starting from $LK_i^{(t)}$. Let $g_1, \ldots, g_d : \{0,1\}^\ell \to \{0,1\}^\ell$ be one-way permutations and $h = h_0$ in $(p\,;c)$-π. Given k with $ic^t \leq k \leq (i+c-1)c^t$, $LK_{i,k}^{(t)}$ is defined by

$$LK_{i,k}^{(t)} = h^{e_0} g_1^{e_1} \cdots g_t^{e_t}(LK_i^{(t)}) \tag{3}$$

where $k - ic^t = e_t c^t + e_{t-1}c^{t-1} + \cdots + e_1 t + e_0$ $(0 \leq e_i < c)$ is a c-ary expansion of $k - ic^t$.

Let us consider the layered keys for the user u_k in the t-th layer. Assume $k = e_t c^t + \cdots + e_1 c + e_0$ for $0 \leq e_0, e_1, \ldots, e_{t-1} < c$ and $e_t \geq 0$. Then the center takes j with $e_t + 1 - (c-1) \leq j \leq e_t + 1$ and gives to the user u_k all the user keys $LK_{j;k_\tau}$ where $k_0 = e_0$ and $k_\tau = \lfloor (\frac{k}{c^\tau} + 1) \rfloor c^\tau$ for $1 \leq \tau \leq t$.

The center assigns these keys to the user u_k along with the interval keys for the scheme $(p\,;c)$-π. Hence the total number of keys for each user is

$$SS_{(p\,;c)} + \sum_{t=1}^{d} \{(c-1)(t+1)+1\} = SS_{(p\,;c)} + \frac{cd(d+3) - d(d+1)}{2}.$$

Encryption/Decryption. If there is no layered interval consisting of all non-revoked users, the center encrypts the session key just as in the scheme $(p\,;c)$-π. Otherwise, we can save the transmission overhead by using layered keys. First the center marks all the layered intervals at each layer which has at least one revoked user as revoked intervals. Next, it finds the leftmost non-revoked interval, say $LP_i^{(d)}$, in the d-th layer. Then the session key is encrypted by $LK_{i,k}^{(d)}$, where u_{k+1} is the first revoked user after u_{ic^d} with $k \leq (i+c)c^d$. The center then marks all the users from $u_{(i-1)c^t+1}$ to u_k and the layered intervals containing at least one of them revoked. This process is repeated for the next non-revoked interval. If there is no non-revoked interval in the d-th layer, go to $(d-1)$-st layer and repeat the same procedure and so on. Finally, if all layered intervals at each layer are revoked, then the scheme $(p\,;c)$-π is applied for the remaining non-revoked users.

Note that each non-revoked user u_k can decrypt the session key by an interval key of $(p\,;c)$-π or a layered key. In order to obtain the key (to decrypt the session key) it costs at most $c-1$ and $t(c-1)$ computations of one-way permutations, respectively. Hence the computation cost is at most $d(c-1)$ computations of one-way permutations.

Transmission Overhead. First we estimate the transmission overhead for $(p\,;c)$-π_1. If there is no revoked user, then $\lceil \frac{N}{c^2} \rceil$ layered intervals cover entire straight line L. By inserting one revoked user to an interval, the interval is divided to at most 3 intervals including punctured or long intervals. So the transmission overhead is at most $\lceil \frac{N}{c^2} \rceil + 2r$. Trivially, the transmission overhead of this scheme cannot be larger than that of punctured interval scheme. So we can conclude that the transmission overhead is at most $\lceil \frac{N}{c^2} \rceil + 2r$ for $r \leq \frac{2}{3} \frac{N(c-1)}{c(c+1)}$ or $\lceil \frac{r}{2} \rceil + \left\lceil \frac{N - \lceil 3r/2 \rceil}{c} \right\rceil$ otherwise. The transmission overhead of $(p\,;c)$-π_d for $d \geq 2$ can be similarly estimated.

3 Revocation Scheme with a Ring Structure

In this section, we present revocation schemes using combination of (punctured) one-way chains and ring structures. We can further reduce user key storage using a hierarchical ring structure. In the following schemes we use relatively simple key assignment applying one permutation to a random label instead of several one-way permutations.

3.1 Revocation Scheme with a Simple Ring Structure

Initially we assume that N nodes (users) are arranged on a ring in a clockwise direction. We denote by u_i the i-th node from the initial node in a clockwise direction and identify u_i and u_j if and only if $i \equiv j \pmod N$. For two nodes u_i and u_j, we set $S_{[i,j]} = \{u_i, u_{i+1}, \cdots, u_{j-1}, u_j\}$. Let $\mathcal{C}_{[i,j]}$ denote a *one-way chain* consisting of all users in $S_{[i,j]}$ that starts from u_i and ends at u_j. For a given one-way permutation $h : \{0,1\}^\ell \to \{0,1\}^\ell$ and an input value $\mathsf{sd} \in \{0,1\}^\ell$, the *chain-value* of $\mathcal{C}_{[i,j]}$ is defined by the value $h^k(\mathsf{sd})$, which is computed by applying h to sd iteratively $k(=j-i+1 \mod N)$ times.

Key Assignment. First, to each node u_i on the ring, a random and independent label $\mathsf{L}_i \in \{0,1\}^\ell$ is selected and assigned. Then the center computes $h^k(\mathsf{L}_i)$, $(1 \leq k \leq N-1)$ and assigns $h^k(\mathsf{L}_i)$ to each node u_{i+k}. Finally the center provides the user u_t with a set of N keys,

$$\{\mathsf{GSK}_0, h^1(\mathsf{L}_{t-1}), h^2(\mathsf{L}_{t-2}), \cdots, h^{N-1}(\mathsf{L}_{t-(N-1)=t+1 \mod N})\},$$

where GSK_0 is an initial group session key.

Encryption. For a given set of revoked users $R = \{u_{i_1}, u_{i_2}, \cdots, u_{i_r}\}$, the center partitions the remaining legal users into r subsets $S_{[i_1+1,i_2-1]}$, $S_{[i_2+1,i_3-1]}$, $\cdots$, $S_{[i_{r-1}+1,i_r-1]}$, $S_{[i_r+1,i_1-1]}$. If u_{i_k} and $u_{i_{k+1}}$ in R are adjacent on the ring, then there is no privileged user between u_{i_k} and $u_{i_{k+1}}$ and the subset $S_{[i_k+1,i_{k+1}-1]}$ is empty. For example, if four users u_3,u_6,u_7,u_{11} are revoked, the set of remaining privileged users is partitioned into 3 non-empty subsets $S_{[4,5]}$, $S_{[8,11]}$ and $S_{[12,2]}$ (see Fig. 4). For each non-empty subset $S_{[i,j]}$, the center assigns a one-way chain $C_{[i,j]}$ and computes its chain-value $h^k(\mathsf{L}_{i-1})$ where $k=j-i+1 \mod N$. Because

there are r arcs in the worst case and a one-way chain is assigned to each arc, the transmission overhead is r.

Decryption. Upon receipt of a broadcast message, each privileged user u_t finds a subset $S_{[i,j]}$ including u_t from the information of indices of revoked users. We note that the subset including u_t is unique. To find the subset $S_{[i,j]}$, u_t performs a binary search on the sequence of indices of the revoked users. Then, by using a value $\mathsf{KV}=h^{(t-i+1) \bmod N}(\mathsf{L}_{i-1})$ given initially, u_t computes a key $h^{j-i+1 \bmod N}(\mathsf{L}_{i-1})$ by applying h to KV $(j-t \mod N)$ times. Each user should compute function h, in the worst case, $N-1$ times.

Basically revoked users are cannot obtain useful any information to decrypt the encrypted session key because of one-wayness of h. However, we should show that the basic scheme is resilient to collusion of any set of revoked users. We can show that the security of the basic scheme is as strong as that of the SD method in [13] by using the following lemma.

Lemma 1. *The above key assignment satisfies the* key-indistinguishability *condition [13] under the pseudo-randomness of a given function h.*

By using a standard hybrid argument on the length of one-way chains we can prove the lemma under the pseudo-randomness of a given function h as in [13]. We omit the proof here.

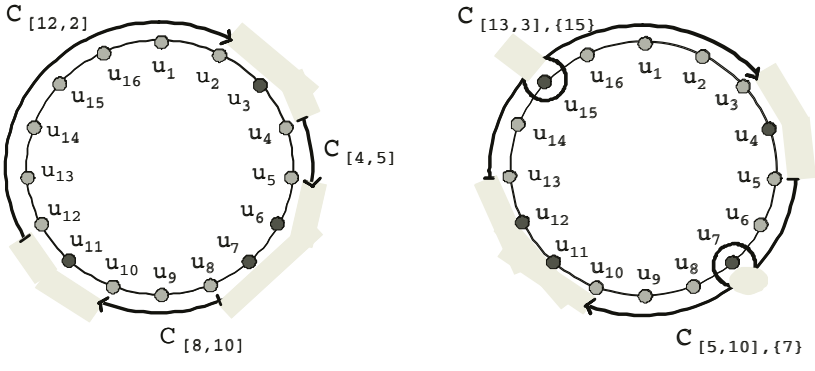

Fig. 4. Revocation in $\mathsf{OFBE}(N{:}1)$ and $\mathsf{OFBE}(N{:}2)$ for $N{=}16$

Now we present a method to further reduce the number of transmission messages in the basic scheme below r. The basic idea is to cover several subsets of privileged users separated by revoked users in the basic scheme by using only one key. This makes the number of messages transmitted drastically goes below r. For a set of users D, let $S_{[i,j],D}$ denote a difference set $S_{[i,j]} \backslash D$, i.e., $\{u \mid u \in S_{[i,j]}$ and $u \notin D\}$. We define a 'jumping' one-way chain $\mathcal{C}_{[i,j],D}$ for $S_{[i,j],D}$ such that, for two nodes u_i and u_j $(u_i \neq u_j)$ and a subset $D=\{u_{k_1},...,u_{k_s}\}$ of $S_{[i,j]}$, it starts from u_i, proceeds in a clockwise direction, but jumps over the nodes $u_{k_1},...,u_{k_s}$, and ends at u_j. First we concentrate on the case that D contains only one revoked user.

Key Assignment. To provide a user with an initial key-set, the center performs the followings : First the center performs the key assignment in the basic scheme. Next, for every pair of (potential revoked) users u_i and u_j not adjacent each other on the ring, the center additionally chooses a random and independent label $\mathsf{L}_{i,j}$ $(1 \le i < j \le N)$. We exclude the cases that u_i and u_j are adjacent on the ring since those cases are covered by one-way chains of the basic scheme. Then the center computes chain-values $h^k(\mathsf{L}_{i,j})$, $(1 \le k \le N-1)$ and assigns it to each node u_{i+k}, $(1 \le k \le N-1)$ except u_i and u_j. In this case, the total number of the keys which a user should store is $N + \binom{N-1}{2} - (N-2) = \binom{N-1}{2} + 2 = \frac{(N-1)(N-2)}{2} + 2$ since the number of keys assigned by the basic scheme is N, the number of cases choosing two different users in the remaining $N-1$ users is $\binom{N-1}{2}$, and the number of cases covered by keys assigned in the basic scheme is $N-2$.

Encryption. To revoke users, the center constructs one-way chains as follows. Starting from any remaining user in a clockwise direction on the ring, the first one-way chain proceeds until it meets the first revoked user u_{i_1}. If the next revoked users u_{i_2} in a clockwise direction is adjacent to u_{i_1}, the chain ends at u_{i_1-1}, just before u_{i_1}. Otherwise the chain jumps over u_{i_1} and continues until it meets u_{i_2}, and ends at u_{i_2-1}, just before u_{i_2}. This process is repeated until all remaining users are covered by one-way chains. For example as in Fig. 4, suppose that users u_4, u_7, u_{11}, u_{12}, and u_{15} are to be revoked. Starting from u_5, the first one-way chain proceeds in clockwise directions, jumps u_7, and ends at u_{10}. The second one-way chain starts at u_{13}, jumps over u_{15}, and ends at u_3. Hence all remaining users are covered by two chains in this example.

By applying the method for r revoked users $\{u_{i_1},...,u_{i_r}\}$, the center broadcasts at most $\lfloor \frac{1}{2} \cdot r \rfloor + 1$ encrypted keys where $\lfloor \cdot \rfloor$ is a floor function. If r is even, $\frac{1}{2} \cdot r$ one-way chains sufficiently cover the remaining privileged users at worst case. If r is odd then we need to cover a last subset $S_{[u_{i_r+1}, u_{i_1-1}]}$, which is not covered by directly assigning 'jumping' one-way chain. In this case, we use a one-way chain $\mathcal{C}_{[i_r+1, i_1-1]}$ of the basic scheme.

Extension. The above cover strategy can be generalized, i.e., naturally extended to cover k subsets by only one key. We denote this method by $\mathsf{OFBE}(N{:}k)$ where N is the number of users.

In $\mathsf{OFBE}(N{:}k)$, the cover strategy is similar to that of $\mathsf{OFBE}(N{:}2)$. Starting from any remaining user u in a clockwise direction on the ring, the center jumps $k-1$ revoked users until it meets the k-th revoked user u_{i_k}. Next, the center goes back in counterclockwise direction to find the first remaining user u'. The first one-way chain starts from u and ends at u', jumping revoked users between u and u' in a clockwise direction. This process is continued until all remaining users are covered. All remaining users are covered by at most $\lfloor \frac{1}{k} \cdot r \rfloor + 1$ chains.

To assign an initial key-set to each user, the center generates a label L_A for every subset $A = \{u_{i_1},...,u_{i_k}\}$ with k users. Then the center computes $h^t(\mathsf{L}_A)$, $(1 \le t \le N-1)$ and gives it to node u_{i_1+t} not in A. To avoid double key assignment, we exclude the cases which can be covered by a key generated in $\mathsf{OFBE}(N{:}j)$ where $0 < j < k$. Hence the number of possible selections is $\mathsf{Num}(N,k)$

$= \binom{N-1}{k} - \binom{N-1}{k-1} + 1$ where $\binom{N-1}{0} = 0$. In $\mathsf{OFBE}(N{:}k)$, a key assignment method is performed recursively. Hence the number of keys which a user should store in $\mathsf{OFBE}(N{:}k)$ is $\#\mathsf{OFBE}(N{:}k) = \#\mathsf{OFBE}(N{:}k-1) + \mathsf{Num}(N,k) = \binom{N-1}{k} + k$ where $\#\mathsf{OFBE}(N{:}j)$ is the number of keys for each user in $\mathsf{OFBE}(N{:}j)$.

3.2 Revocation Scheme with a Hierarchical Ring Structure

In this subsection we generalize the basic scheme $\mathsf{OFBE}(N{:}1)$ by extending a simple ring structure to a hierarchical one. A *hierarchical ring* R is recursively defined such that there is an outmost ring R_0 in level 1 having nodes on the ring, each node on R_0 has children arranged on a ring in level 2, and each node in level 2 again has children on a ring in level 3, and so on. A node which does not have any child is called a *leaf*. The *depth* of a hierarchical ring is the highest level of a leaf in the ring. A hierarchical ring R is called *w-ary* if each node in R has w children.

Underlying Scheme of Our Generalization. For our generalization, we first describe an underlying revocation scheme with a relatively simple hierarchical ring R with depth 2 such that there are two nodes u_1 and u_2 in level 0 (i.e., on the outmost ring). Each u_b for $b \in \{1,2\}$ has m children $\{u_{b.1}, .., u_{b.m}\}$ on a sub-ring R_b, where $u_{b.j}$ denotes the j-th user from $u_{b.1}$ on ring R_b. Hence the number of users in R are $2 \cdot m$. We denote this scheme by $\mathsf{HOC}(2, m)$.

Let $S_{\bar{b},[b.i,b.j]}$ denote a union set $\mathsf{R}_{\bar{b}} \cup S_{[b.i,b.j]}$, i.e., $\{u | u \in S_{[b.i,b.j]}$ and $u \in \mathsf{R}_{\bar{b}}\}$ where $\bar{b} = 2^{(b \mod 2)}$. We define a simple 'hierarchical' one-way chain $\mathcal{C}_{\bar{b},[b.i,b.j]}$ for $S_{\bar{b},[b.i,b.j]}$ such that it starts from a node $u_{\bar{b}}$, goes through children $u_{b.i}, u_{b.i+1}$,..., $u_{b.j-1}$ of u_b in a clockwise direction and ends at $u_{b.j}$ on R_b. $\mathcal{C}_{\bar{b},[b.i,b.j]}$ is used to cover users on two separated rings by one key, i.e., all users on $\mathsf{R}_{\bar{b}}$ and some users on R_b. For example as in Fig. 5, suppose that $u_{1.6}$, $u_{1.11}$, and $u_{1.15}$ on R_1 are to be revoked. the center partitions the remaining privileged users into disjoint subsets $\mathsf{R}_2 \cup S_{[1.7,1.10]}$, $S_{[1.12,1.14]}$ and $S_{[1.16,1.5]}$. Then the center assigns one-way chains $\mathcal{C}_{2,[1.7,1.10]}$, $\mathcal{C}_{[1.12,1.14]}$, $\mathcal{C}_{[1.16,1.5]}$ to those subsets, respectively.

Key Assignment. To exploit the previous hierarchical cover strategy using two types of one-way chains, two types of labels are selected by the center initially: One is a label used in the basic scheme. The other is a label L_b associated to a node v_b, which is used to compute a chain-value corresponding to a hierarchical one-way chain. In this case, we use a cryptographic pseudo-random sequence generator $G{:}\{0,1\}^{\ell} \rightarrow \{0,1\}^{m\ell}$. We denote by $G(\mathsf{L})_i$ the i-th output block of length ℓ of G on L. A similar function which triples an input is used in the SD scheme [13] to achieve a similar purpose.

The center provides a user $u_{b.t}$ with n keys by using the following key assignment method: First the center performs the key assignment of the basic scheme to cover only privileged users in one sub-ring of m users. That is, for each $b \in \{1,2\}$ and $u_{b.i} \in \mathsf{R}_b$, a label $\mathsf{L}_{b.i} \in \{0,1\}^{\kappa}$ is selected. Then the center computes $h^k(\mathsf{L}_{b.i})$, $(1 \leq k \leq m-1)$ and assigns it to $u_{b.i+k}$. Next, for each $b \in \{1,2\}$, a label $\mathsf{L}_b \in \{0,1\}^{\ell}$ is selected. Then the center computes $h^k(G(h(\mathsf{L}_b))_i)$, $(1 \leq k \leq m-1)$ and assigns it to $u_{\bar{b}.i+k}$. Finally the center provides $u_{b.t}$ with a set of the following keys;

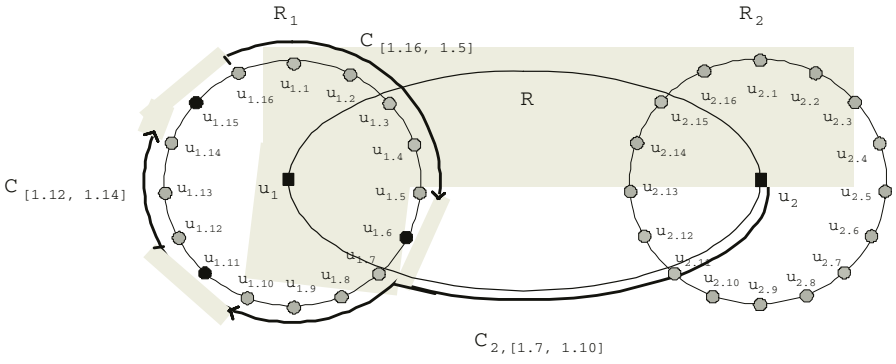

Fig. 5. Revocation in $\mathsf{HOC}(2, m)$

$$\{\ \mathsf{GSK}_0,\ h(\mathsf{L}_{b.j-1}), h^2(\mathsf{L}_{b.j-2}), \cdots, h^{m-1}(\mathsf{L}_{b.(j-(m-1)=j+1 \mod m)}),$$
$$h(\mathsf{L}_{\bar{b}}), h(G(h(\mathsf{L}_b))_{j-1}), h^2(G(h(\mathsf{L}_b))_{j-2}), \cdots, h^{m-1}(G(h(\mathsf{L}_b))_{j-(m-1)=j+1 \bmod m})\}$$

Encryption. To revoke users, the center covers the remaining legal users as follows: If all revoked users are included in only one sub-ring R_b, the center generates one-way chains $\mathcal{C}_{\bar{b},[b.i_1+1,b.i_2-1]}$ and $\{\mathcal{C}_{[b.i_2+1,b.i_3-1]}, \cdots, \mathcal{C}_{[b.i_{r_b}+1,b.i_1-1]}\}$. Otherwise, the center performs the revocation method of the basic scheme on each sub-ring. Then the center encrypts a group session key GSK by using chain-values as keys and broadcasts the ciphertexts. In particular, for a chain-value of the hierarchical one-way chain $\mathcal{C}_{\bar{b},[b.i,b.j]}$, the center computes $h^{j-i+1 \mod m}(G(h(\mathsf{L}_b))_{i-1})$. The number of messages to be transmitted is at most r as in the basic scheme.

Security. The security of this scheme depends on the association of the securities of two functions, h and G. However, without considering the association, we can use one function instead of two independent functions. In this case, for the expansion of labels, we can define the output of h as the first output block of G, namely, $h(\mathsf{L})=G(\mathsf{L})_1$ for $\mathsf{L}\in\{0,1\}^\ell$. Using the similar idea in [13] we can prove the security under the pseudo-randomness of G.

Generalization. Naturally, we can extend the previous $\mathsf{HOC}(2, m)$ by allowing more levels and more nodes in each level. In general schemes the center uses a hierarchical one-way chain traversing nodes in many levels, which starts from a node of level d and goes through some nodes of the same level in a clockwise direction and ends at u_j, and comes down to a children node of u_{j+1} in level $d+1$ and iterates this process. For space limitation, we omit the concrete explanation of the general scheme.

One important thing is that there is a trade-off relation between transmission overhead and keys stored at a user. Using the deepest hierarchical ring structure such as a complete binary ring of depth $\log_2 n$, we gain reduction in user storage up to $\frac{\log_2^2 n+\log_2 n}{2}+1$ while the transmission overhead increases up to $2r$. Interestingly, the revocation scheme for a binary (2-way) ring is structurally

equivalent to SD based on a binary tree proposed by Naor et. al [13]. The forms of subsets to be covered in both schemes are equivalent, since a binary ring is structurally equivalent to a binary tree.

To further reduce the transmission overhead, we can use 'jumping' hierarchical one-way chains which jump potential revoked users and cover several subsets of privileged users separated by revoked users at a time. Our generalization provides a generic method for any application in terms of setting specific values for parameters to achieve a desirable trade-off.

3.3 Specific Revocation Scheme

In this section we present a specific revocation scheme where the transmission overhead is less than r while storage requirement per user is reasonably low. First we notice that different $\mathsf{OFBE}(N{:}k)$ schemes can be independently combined with revocation schemes using hierarchical one-way chains. In particular $\mathsf{OFBE}(m{:}2)$ can be directly applied to each sub-ring R_b of $\mathsf{HOC}(2, m)$, which contains $m{=}\frac{N}{2}$ users. In this case, the number of keys which a user should store reduces from $\frac{N^2-3N+6}{2}$ to $\frac{m^2-m+6}{2}$ while the transmission overhead is still at most $\lfloor \frac{1}{2} \cdot r \rfloor{+}1$. We denote this method by $\mathsf{HOC}(2{,}[m{:}2])$.

Lemma 2. *In HOC(2,[m:2]), the number of keys stored by a user is* $\frac{m^2-m+6}{2}$ *and the transmission overhead is at most* $\lfloor \frac{1}{2} \cdot r \rfloor{+}1$*, for* $N = 2m$*.*

Though, in $\mathsf{HOC}(2{,}[m{:}2])$, the number of keys storage per user is reduced while the number of transmission messages is still at most $\lfloor \frac{1}{2} \cdot r \rfloor{+}1$, the method is still not applicable for a large number of group users. To reduce user storage further with slight increase in transmission complexity, we can use divisional approach as follows: First partition a group of N $(=2m{\cdot}s)$ users into s subgroups and then apply $\mathsf{HOC}(2{,}[m{:}2])$ to each sub-group of $2m$ users where m is a predetermined constant. We denote this method by $\mathsf{HOC}(2{,}[m{:}2])_{\mathsf{p}}$. In this case, the transmission overhead is $\lfloor \frac{1}{2} \cdot r \rfloor{+}\frac{N}{2m}$ at worst case and each user should store $\frac{m^2-m+6}{2}$ keys.

4 Comparison

Table 1 shows the complexity of the storage sizes, the transmission overhead and the computation costs of our schemes, SD and LSD when $N = 10^8$ and r is $0.1, 0.5, 1, 5, 10$ and 20% of N. In the table, we assume that the size of a user key is 128 bits, which is considered reasonably secure, currently.

Figure 6 shows the comparison of the *worst-case* transmission overheads by graphs when the revocation rate ranges from 0% to 3%. Among the graphs, the dotted line represents the transmission overhead of the scheme $(1\,;100)$-π_1. The dotted graph is very close to that of SD for small r. It has steeper slope than the graph of $(1\,;100)$-π, but a lower y-intercept at $\lceil \frac{N}{c} \rceil$. As we mentioned above, the layered π scheme improves the transmission overhead when the revocation rate is small. For large r, it has the same transmission overhead as that of the scheme $(p\,;c)$-π.

Table 1. Examples when $N = 10^8$

Scheme	Storage	TO (Mbits)						CC
r revoked	(KBytes)	0.1%	0.5%	1%	5%	10%	20%	
$(0\,;100)$-π	1.60	141	191	253	755	1380	2640	100
$(1\,;100)$-π	79.2	134	159	190	438	749	1370	100
$(0\,;100)$-π_1	4.80	26.9	129	253	755	1380	2640	198
$(1\,;100)$-π_1	82.4	26.9	129	190	438	749	1370	198
$\mathsf{HOC}(2,[100:2])_{\mathsf{p}}$	79.2	70.4	96	128	384	704	1344	100
$\mathsf{HOC}(2,[50:2])_{\mathsf{p}}$	19.6	134.4	160	192	448	768	1408	50
SD	5.8	25.6	128	256	1280	2560	5120	27
LSD	2.3	51.2	256	512	2560	5120	10240	27

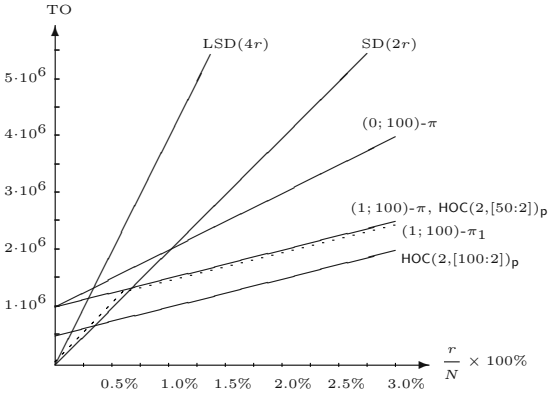

Fig. 6. TO for $N = 1 \cdot 10^8$ in the worst case

The transmission overhead of $\mathsf{HOC}(2,[50{:}2])_{\mathsf{p}}$ or $\mathsf{HOC}(2,[100{:}2])_{\mathsf{p}}$ is relatively larger than that of SD at initial interval where the number r of revoked users is smaller than 0.75 % of the total users. But, except this interval, the transmission overhead of $\mathsf{HOC}(2,[50{:}2])_{\mathsf{p}}$ becomes, at worst case, about $\frac{1}{3.5}$ of the transmission overhead of SD. This should be a good trade-off in most applications since the number of initial messages is relatively small. The number of keys per user in $\mathsf{HOC}(2,[50{:}2])_{\mathsf{p}}$ is about 3.5 times as many as that of SD as the number of revoked users increases. But this difference may be acceptable in many practical applications.

5 Conclusion

In this paper, we proposed efficient broadcast encryption schemes based on linear and circular structures. Introducing the idea of punctured one-way chain to these strucutres, we could reduce the transmission overhead below r. Particu-

larly, our specific schemes have about 1/3 transmission overhead than SD while maintaining the computation cost and the storage size in a reasonable bound.

Moreover our methods provide many flexibility on the system efficiency. The system can be optimized to have best efficiency for any of the three parameters of broadcast encryption the transmission overhead, the computation cost and the storage size.

References

1. J. Anzai, N. Matsuzaki and T. Matsumoto, *A quick key distribution scheme with "Entity Revocation"*, Advances in Cryptology - Asiacrypt'99, Lecture Notes in Computer Science 1716, pp.333-347.
2. S. Berkovits, *How to Broadcast a secret*, Advances in Cryptology - Eurocrypt'91, Lecture Notes in Computer Science 547, pp.536-541.
3. D. Boneh and A. Silverberg, *Applications of Multilinear Forms to Cryptography*, Contemporary Mathematics 324, American Mathematical Society, pp.71-90.
4. B. Chor, A. Fiat and M. Noar, *Tracing Traitors*, Advances in Cryptology CRYPTO'94, Lecture Notes in Computer Science 839, pp. 257-270.
5. G. Chick and S. Tavares, *Flexible access control with master keys*, Advances in Cryptology - Crypto'89, Lecture Notes in Computer Science, pp.316-322.
6. P. D'Aroco and D.R. Stinson, *Fault Tolerant and Distributed Broadcast Encrytion*, CT - RSA'03, Lecture Notes in Computer Science 2612, pp.263-280.
7. A. Fiat and M. Naor, *Broadcast Encryption*, Advances in Cryptology - Crypto'93, Lecture Notes in Computer Science 773, pp.480-491.
8. M.T. Goodrich, J.Z. Sun and R. Tamassia, *Efficient Tree-Based Revocation in Groups of Low-State Devices*, Advances in Cryptology - Crypto'04, Lecture Notes in Computer Science 3152, pp.511-527.
9. J. Garay, J. Staddon and A. Wool, *Long-Lived Broadcast Encryption*, Advances in Cryptology - Crypto'00, Lecture Notes in Computer Science 1880, pp.333-352.
10. E. Gafni, J. Staddon and Y.L. Yin, *Efficient Methods for Integrating Traceability and Broadcast Encryption*, Advances in Cryptology - CRYPTO'99, Lecture Notes in Computer Science 1666, pp.372-387.
11. D. Halevi and A. Shamir, *The LSD Broadcast Encryption Scheme*, Advances in Crytology - Crypto'02, Lecture Notes in Computer Science 2442, pp.47-60.
12. R. Kumar, S. Rajagopalan and A. Sahai, *Coding Constructions for blacklisting problems without Computational Assumptions*, Advances in Cryptology - Crypto'99, Lecture Notes in Computer Science 1666, pp.609-623.
13. D. Naor, M. Naor and J. Lotspiech, *Revocation and Tracing Schemes for Stateless Receivers*, Advances in Cryptology - Crypto'01, Lecture Notes in Computer Science 2139, pp.41-62.
14. M. Naor and B. Pinkas, *Efficient Trace and Revoke Schemes*, Financial Cryptography'00, Lecture Notes in Computer Science.
15. C.K. Wong, M. Gouda and S.S. Lam, *Secure Group Communication using Key Graphs*, ACM SIGGCOM'98 ACM.
16. M. Luby and J. Staddon, *Combinatorial Bounds for Broadcast Encryption*, Advances in Cryptology - Eurocrypt'98, Lecture Notes in Computer Science 1403, pp.512-526.

Author Index